GEOMETRY COMPRESSION OF ISOMORPHIC AND BLOCK-ISOMORPHIC 3D ANIMATIONS

Sanjib Das

GEOMETRY COMPRESSION OF ISOMORPHIC AND BLOCK-ISOMORPHIC 3D ANIMATIONS

A

Thesis submitted

for the award of the degree of

DOCTOR OF PHILOSOPHY

By

Sanjib Das

DEPARTMENT OF ELECTRONICS AND ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
GUWAHATI - 781 039, INDIA

November 2019

Dedicated

to ***My Parents***

for their blessings...

and

to ***My Family***

for their love and encouragement...

Certificate

This is to certify that the thesis entitled "**Geometry Compression of Isomorphic and Block-isomorphic 3D Animations**", submitted by **Sanjib Das** (06610214), a research scholar in the *Department of Electronics and Electrical Engineering, Indian Institute of Technology Guwahati*, for the award of the degree of **Doctor of Philosophy**, is a record of an original research work carried out by him under our supervision and guidance. The thesis has fulfilled all requirements as per the regulations of the Institute and in our opinion has reached the standard needed for submission. The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Dr. P. K. Bora
Professor

Dr. A. K. Gogoi
Professor

Department of Electronics and Electrical Engineering
Indian Institute of Technology Guwahati
Guwahati - 781 039, India.

Dated:

Place: IIT Guwahati

Acknowledgements

At the outset, I would like to express my sincere gratitude and wholehearted thanks to my thesis supervisors, Prof. Prabin Kumar Bora, and Prof. Anup Kumar Gogoi, for their constant support, excellent guidance and encouragement throughout my research work. I greatly admire their attitude towards research, creative thinking, hard work and dedication in work. I am indebted to Prof. P. K. Bora for his advice and keen attention to minor details in my research which motivated me in all the time of research and writing of this thesis work. I must acknowledge his kindness, patience and extreme diligence in correcting all my manuscripts and thesis.

Besides my supervisors, I wish to extend my sincere gratitude to the members of my doctoral committee, Prof. Samarendra Dandapat, Prof. Chitralekha Mahanta and Dr. Tony Jacob for sparing their precious time to evaluate the progress of my work. My heartfelt thanks to them for their insightful comments, encouragement and continuous feedback for improving my research from various perspectives. I am also thankful to my earlier doctoral committee member, Prof. Anil Mahanta, for his constructive suggestions, advice, and encouragement to carry out this research work. Many faculty members of the department gave me suggestions and encouragement from time to time. Their kind support and valuable suggestions helped me a lot to improve myself academically and personally. I sincerely acknowledge and thank the faculty members in this regard.

I would like to acknowledge the help and support shown by my departmental colleagues, Dr. L. N. Sharma, Mr. P. J. Goswami, Mr. Utpal Sarma, Dr. M. P. Das, Ms. Josephine S., Mr. S. Sonowal, Mr. D. Gogoi, Ms. R. Rabha, Mr. R. Bharali, Mr. S. S. Mazid, Mr. Mukut Baruah, Mr. Dasarath Das, Mr. N. J. Dutta, Mr. T. Kakati, Ms. Sankari Dutta and Late. Uday S. Uzir, during the tenure of my research work. My special thanks go to Dr. L. N. Sharma for encouraging me with the valuable discussions that we had time to time and also helping me with his constructive suggestions during writing of my final thesis. I would also like to thank all the newly joined technical staff of the department, Paban, Abhishek, Jatin, Sumit, Sabita, Chayanika, Phool Chand, Motiur, Rakesh and Khursida for their help.

I would like to thank all research scholars of the EEE department who helped me directly or indirectly during my research work. My special thanks go to all the research scholars of Electro-

medical and Speech Technology Laboratory for their support and presence during the writing of my final thesis.

I am also grateful to the staff of the Academic Section at IIT Guwahati for their kind help and assistance at all times throughout my Ph.D. tenure.

I would also like the acknowledge the moral support and encouragement received from Dr. Sabita Baruah (Madam), Mrs. Pramila Gogoi (Madam), Mrs. Indira Sarma, Ms. Anisha Sharma and Mrs. Purabi Sonowal during my research journey.

Last but not least, I would like to express my sincere thanks and deepest gratitude to my family members for their love and support during the entire period of my research. My greatest debt of gratitude is owed to my parents, whose hard work and sacrifices gave me the opportunities I have today. I would also like to sincerely acknowledge the love, affection, encouragement, and moral support received from my wife 'Debjani', son 'Snehashis', sister 'Sampa', brother 'Sanjoy', sister-in-law 'Samarpita', nephews 'Piyush and Aayush' and cousin brother 'Biswajit' towards the completion of this research work.

Finally, I am grateful to Almighty for giving me enough strength, patience, and courage to overcome all the challenges and have guided me through the righteous path to complete this thesis work successfully.

Sanjib Das

Abstract

The large volume of raw animation geometry data of complex 3D models demands for the efficient compression of these data. The most of the existing animation geometry compression methods consider the 3D animation sequences comprising a single object with equal number of vertices per frame and similar connectivity across the full animation frames. Such an animation is called an *isomorphic single-object animation* and it is characterised by the changes in geometry only. However, in real animations, the geometry as well as the connectivity of vertices in the 3D mesh sequence may change over the frames. In a particular case, the animation may comprise isomorphic objects in blocks of frames. The research work presented in this thesis centres around the development of efficient algorithms for the compression of isomorphic and block-isomorphic animation geometry data.

One of the important animation geometry compression methods is the clustered PCA (CPCA) method which clusters the 3D vertex trajectories on the basis of motion similarity and applies the PCA-based compression on each cluster. An algorithm is proposed for improving the CPCA method. The proposed subtractive clustering based clustered PCA (S-CPCA) method provides a stable initialization of the cluster centres by applying the density function based subtractive clustering on the vertex trajectories. It also includes improved bit allocation and a block-based arithmetic coding scheme for the compression parameters. The proposed S-CPCA based method outperforms the other PCA based methods, namely the CPCA and the LPCA based methods in terms of the KG_{err} and the $STED$ distortion metrics.

Scalability of the animation geometry compression algorithm is a challenging issue. A novel scalable compression method is proposed to obtain spatio-temporal scalability using the singular value decomposition (SVD) of the vertex trajectory matrix of the geometry data. The set of elements in the spatial and temporal singular vectors are decomposed into

different spatio-temporal layers to support scalability. The prediction of elements in all the upper spatio-temporal layers is performed using the averaging of neighbourhood elements in the lower layers. The prediction errors of each spatio-temporal layer are quantized using a non-uniform bit allocation scheme and entropy coded using a block based arithmetic coding. The proposed compression algorithm outperforms existing methods in terms of the scalable rates and distortions.

The compression of multi-object block isomorphic frames is not yet explored. The thesis proposes a novel technique for the compression of block isomorphic multi-object (BIMO) 3D animation geometry. The proposed method applies a temporal segmentation algorithm to divide the animation data of the consecutive frames into different isomorphic multi-object (IMO) blocks. A proposed connectivity based spatial segmentation algorithm is applied on each IMO block to detect the vertices belonging to the individual 3D objects. An object based PCA (OPCA) algorithm is applied on the vertex trajectories of each 3D object to get the compression. The proposed method shows very good compression performance on the block isomorphic animation geometry data.

Developing a perceptual visual distortion metric for the quality assessment of the decompressed 3D animation data is an important research issue. The thesis proposes an improved perceptual based visual distortion metric called the affine invariant spatio-temporal edge difference (AISTED) metric. It first finds the affine-invariant representations of the reconstructed frames using a least-squares method. The metric is then computed using the relative changes in spatial and temporal edge lengths between the original frames and the corresponding affine-invariant representations. The proposed metric shows improved correlation with the subjective assessment scores in terms of the Pearson's linear correlation coefficient (PLCC) and the the Spearman's rank order correlation coefficient (SROCC).

Contents

List of Figures

List of Tables

List of Acronyms

2D	Two Dimensional
3D	Three Dimensional
3DMC	3D mesh coding
4D	Four Dimensional
AISTED	Affine invariant spatio temporal edge difference
AFX	Animation Framework eXtension
AHD	Average Hausdroff distance
AMRMSE	Average maximum root mean square error
APD	Angle-based perceptual distance
BIMO	Block-isomorphic multi-object
BISO	Block-isomorphic single-object
CC	Connected component
CABAC	Context adaptive binary arithmetic coding
CODDYAC	Connectivity driven dynamic mesh compression
CPCA	Clustered principal component analysis
DAME	Dihedral angle mesh error
DCT	Discrete cosine transform
DMPD	Dynamic mesh perceptual distance
DWT	Discrete wavelet transform
DM	Distortion metric
DMOS	Differential MOS
EV	Eigen vector

FAMC	Frame-based animated mesh compression
FKM	Fuzzy K-means
FMPD	Fast mesh perceptual distortion
GL	Geometric Laplacian
GOF	Group of frame
HD	Hausdroff distance
HVS	Human visual system
IMO	Isomorphic multi-object
LCPS	Linear coding based on pose similarity
LD	Layer decomposition
LoD	Level of detail
LPC	Linear predictive coding
LPCA	Local principal component analysis
LSP	Least square predictor
MDC	Multi-order differential coding
MOS	Mean opinion score
MO-3D	Multi-object based 3D
MRMSE	Maximum root mean square error
MSDM	Mesh structural distortion measure
MSE	Mean square error
MPEG4	Motion Picture Expert Group 4
MV	Mean vector
NIMO	Non-isomorphic multi-object
OPCA	Object-based principal component analysis
PSNR	Peak signal to noise ratio
PCA	Principal component analysis
PC	Principal component
PLCC	Pearson's linear correlation coefficient
PDF	Probability density function

PVD	Perceptual visual distortion
RBFP	Radial basis function predictor
RD	Rate-distortion
RMSE	Root mean square error
SC	Subtractive clustering
S-CPCA	Subtractive clustering based clustered principal component analysis
SM	Spatial masking
SNR	Signal to noise ratio
SO-3D	Single-object based 3D
SPC	Scalable predictive coding
SPPD	Speed based perceptual distance
SROCC	Spearman rank order correlation coefficient
SVC	Scalable video coding
SVD	Singular value decomposition
SWA	Spatial wavelet analysis
SWSPD	Speed-weighted spatial-based perceptual distance
TC	Transform coefficient
TFAN	Triangle fan
TM	Temporal masking
TPDM	Tensor-based perceptual distortion measure
TWT	Temporal wavelet transform
WI	With-user interaction
WOI	Without-user interaction
VLC	Variable length coding

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati

List of Symbols

$\tilde{(\,)}$	Reconstructed quantity
α_i^f	Angle between forward and backward motion vector
β_{ij}^f	Angle opposite to a edge connecting ${v_i}^f$ and ${v_j}^f$
$\delta a(i,f)$	Angle based perceptual distance
$\delta r(i,f)$	Difference of local roughness at v_i^f and $\tilde{v}_i^f$
$\delta s(i,f)$	Difference of in norm of motion vectors between v_i^f and $\tilde{v}_i^f$
$\varepsilon_{u_{i,l}}^j$	Prediction error for i-th element of l-th spatial layer
$\varepsilon_{w_{i,l}}^j$	Prediction error for i-th element of l-th temporal layer
κ_l	Connectivity data for l-th spatial layer
$\|.\|_f$	Frobenius norm
$\|.\|_2$	Euclidean norm
λ_i	i-th eigen value
λ_{Th}	Threshold value for energy in eigen values
$\mathbf{\Lambda}$	Diagonal matrix with eigen values
$\boldsymbol{\mu}^f$	Mean of vertices of frame f
$\mathbf{\Phi}^j$	Vertex trajectories in cluster j
ρ_p	Pearson's linear correlation coefficient
ρ_s	Spearman's rank order correlation coefficient
σ_i	i^{th} singular value
σ_f^2	Variance of vertices in frame f
$\mathbf{\Sigma_s}$	Diagonal matrix with singular values
$\Psi\,()$	Cost function

θ_k	Angle formed by k-th facet to ${v_i}^f$
$\xi_f^2()$	Mean square error
$\mathbf{a}_f$	Vector represention of geometry data for f-th frame
$\mathbf{A}$	Frame-based animation geometry matrix of size $3N_v \times N_f$
$\mathbf{A}_m$	Mean centreded geometry matrix
$\mathbf{B}$	Trajectory-based animation geometry matrix of size $N_v \times 3N_f$
$bpvf$	Bits per vertex per frame
$\mathbf{C}_s$	Connectivity matrix of size $N_t \times 3$ for static mesh
$\mathbf{C}_d$	Connectivity matrix of size $N_t \times 3$ for dynamic mesh
CR	Compression ratio
$\mathbf{Cov}_A$	Covariance matrix for $\mathbf{A}$
$\mathbf{Cov}_f$	Covariance matrix for vertices in frame f
$d_H(\mathcal{M}_s, \tilde{\mathcal{M}}_s)$	Hausdorff distance between $\mathcal{M}_s$ and $\tilde{\mathcal{M}}_s$
d_{ij}	Euclidean distance between i^{th} and j^{th} trajectories
$\mathbf{d}_i^f$	Forward motion vector at v_i^f
$\mathbf{D_t}$	Distance matrix of trajectories
D_{ij}^f	Laplacian matrix at vetex v_i^f
$d(v_j, \tilde{\mathcal{M}}_s)$	Euclidean distance between a vertex v_j and the closest vertex in mesh $\mathcal{M}_s$
e_{ij}	Spatial edge between vertex ${v_i}^f$ and ${v_j}^f$
$el(e_{ij}, f)$	Length of spatial edge e_{ij} at frame f
$ed(e_{ij}, f)$	Relative edge difference between at frame f
$\mathbf{E}(\mathbf{A})$	Mean frame matrix of size $3N_v \times N_f$
E_s	Set of unique edges
$\mathbf{f}_n$	n-th triangular face
$\mathbf{G}_d$	Geometry data of a dynamic mesh
$\mathbf{G}_s$	Geometry data of a static mesh
$\mathbf{G}^f$	Geometry data of f-th frame of size $N_v \times 3$
GC_i^f	Gaussian curvature at vertex v_i^f
$\mathbf{I}$	Identity Matrix

$\mathbf{I}_c$	Cluster index vector
I_v	Set of vertex indices
KG_{err}	KG Error
$\mathbf{LU}_u^l$	Matrix with elements of spatial singular vectors for l-th layer
$\mathbf{LW}_w^l$	Matrix with elements of temporal singular vectors for l-th layer
$\mathbf{m}_A$	Vector representing mean frame
$\mathcal{M}_s$	Original static mesh
$\mathcal{M}_{s_i}$	Static mesh representing the i-th frame of a dynamic mesh
$\mathcal{M}_d$	Original dynamic mesh
MEE_s	Mean spatial edge error
MEE_t	Mean temporal edge error
N_v	Number of vertices per frame
N_f	Number of frames
N_t	Number of triangular faces
N_e	Number of unique edges
P_i	Potential value at i^{th} vertex trajectory
$\mathbf{p}_{v_i}$	3D coordinates of vertex v_i
$\mathbf{p}_{v_i}^f$	3D coordinates of vertex v_i^f
$\mathbf{p}_{v_i}$	3D coordinates of i-th vertex in a static mesh
q_i	i^{th} quantization bit
r_i^f	Local roughness at v_i^f
$\mathbb{R}$	Set of Real numbers
R_l	Set of vertex patches removed at l-th spatial layer
$\mathbf{R}$	Rotation matrix
σ_j	j^{th} singular value
s_i	i^{th} quantization step size
$s_t(i, f, w)$	Average spatio-temporal speed of v_i^f
$\mathbf{\Sigma_s}$	Diagonal matrix with singular values
$se_{err}(f)$	Normalized spatial edge error for f^{th} frame

$STED$	Spatio-temporal edge difference metric
$tel(i, f)$	Temporal edge length at i^{th} vertex of f^{th} frame
$ted(i, f, w)$	Relative temporal edge difference
$te_{err}(f)$	Normalized temporal edge length error for f^{th} frame
$\mathbf{t}_i$	Vector representing the i^{th} vertex trajectory
$\mathbf{t}_{c_j}$	j^{th} cluster centre trajectory
T_d	Set of triangulations for degree-d patch
$\mathbf{T}$	Homogeneous Transformation matrix
$\mathbf{T_r}$	Inverse Transformation matrix
T_{bpvf}	Target $bpvf$
u_i^j	i^{th} spatial geometry element in $\mathbf{u}_j$
$u_{i,l}^j$	i^{th} element of j^{th} spatial singular vector at layer l
$\mathbf{u}_j$	j^{th} spatial singular vector
$\mathbf{U}$	Matrix with eigen vectors of $\mathbf{Cov}_A$
$\mathbf{U}_c$	Matrix with c eigen vectors
$\mathbf{U}^j$	Matrix with Eigen vectors for cluster j
$\mathbf{U_s}$	Matrix with spatial singular vectors of $\mathbf{B}$
v_i	i^{th} vertex of a mesh
$v_{i_j}^f$	j-coordinate of the vertex v_i
V_l	Set of vertex indices in l^{th} spatial layer
$w_i^{j,l}$	i-th element of j^{th} temporal singular vector at layer l
$w_{f_j}^j$	j component of f^{th} temporal geometry element in $\mathbf{w}_j$
$\mathbf{w}_j$	j^{th} temporal singular vector
$\mathbf{W_s}$	Matrix with temporal singular vectors of $\mathbf{B}$
$W^{j,l}$	Set of elements of $\mathbf{w}_j$ at l^{th} temporal layer
$\mathbf{X}_c$	Set of cluster centre trajectories
$\mathbf{Y}$	PCA transformed coefficient matrix
$\mathbf{Y}^j$	PCA transformed coefficient matrix for cluster j
$\mathbb{Z}^+$	Set of positive integers

1

Introduction

Contents

1.1 Introduction

3D animation is the rapid computer display of 3D objects/models with changing attributes over a time span. It is a sub-field of computer graphics, which has shown rapid development over the last decades. This is mainly because of the recent advancement of the 3D acquisition technology using the high-end computer graphics hardware, the development of the 3D modelling software and the growth of processing power of desktop PCs. The invention of 3D acquisition hardware, such as 3D scanners, 3D cameras, depth sensors, MRI, radar, ultrasound etc. has facilitated users to digitize the real world complex 3D objects. The evolution of 3D modelling software has also enabled people to design virtual 3D models to simulate the real 3D objects. As a result, there is an abundance of 3D model databases which are available for use in the areas of science, engineering, entertainment industries and other fields. With the popularization of multimedia technologies, there is a growing demand to use these 3D models for visualizing and simulating 3D objects in the applications like 3D television, computer generated animated films, 3D video games, engineering and architectural design, interactive scientific simulations, e-commerce, virtual reality and so on. Moreover, the advancement of Internet technologies has made possible for the people to share, distribute and access the 3D model databases more easily. However, these 3D models need a large storage space, large bandwidth for transmission and longer reconstruction time by the graphics hardware for real-time visualization. So, there is an increasing demand for the efficient compression of the 3D models for compact storage, fast transmission and lower reconstruction time.

Researchers have started experiments from early 90s in this direction and developed a number of methods for compression of 3D models. They have also succeeded in standardizing some methods as a part of Motion Picture Expert Group 4 (MPEG4) standard of Part-16 for Animation Framework eXtension (AFX) [2]. Moreover, the present compression methods have to support the heterogeneous client system that needs to facilitate different end users with different hardware capabilities. Since making different bit streams for different end users is not an efficient solution to the problem, there is a need to encode the compressed 3D data with a single bit-stream to enable users with different hardware to derive the required information from the same. This method of obtaining a unique bit stream suitable for a heterogeneous client system has also become important issue. This thesis addresses the issues of scalable and non-scalable compression of the 3D models.

1.2 3D animation basics

This section presents a brief introduction to the acquisition process of 3D models in the form of 3D meshes. It is followed by a discussion about the different attributes needed to represent 3D meshes. The different categories of 3D mesh data and their mathematical representations are also outlined.

1.2.1 3D data acquisition

A simple block diagram for 3D model acquisition process is shown in Figure 1.1.

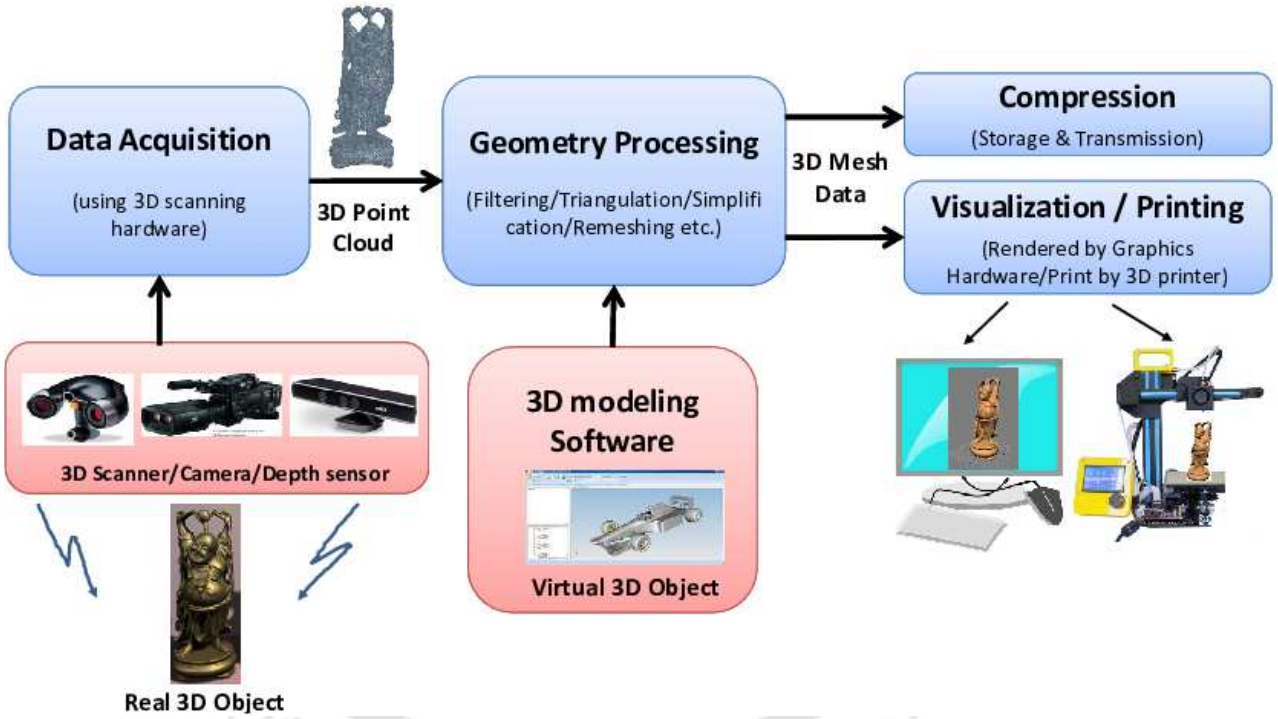


Figure 1.1: An example of 3D Model acquisition process

A 3D scanner or depth field sensor is used to capture the surface of the real world 3D objects in the form of 3D point clouds by laser scanning the object from multiple directions. These 3D point clouds are then passed through geometric processing routines like smoothing, filtering or simplification using the 3D modelling software. Finally, a wireframe representation of the digitized surface of the 3D object is created using the triangulation technique. The triangulation method connects the adjacent vertices on the surface of the model with a set of triangles called *3D triangular meshes*. The 3D virtual models having triangular meshes may also be created using the 3D modelling software from the Two

Dimensional (2D) image of an object. These 3D triangular meshes are then represented in a popular 3D format to enable rendering by the graphics hardware for the visualization of the end-users. The digitized data can also be used for the mass reproduction of the 3D model using the 3D printers. The size of the voluminous 3D mesh data is also compressed using standard compression algorithms for compact storage and faster transmission over network.

1.2.2 Attributes of 3D triangular meshes

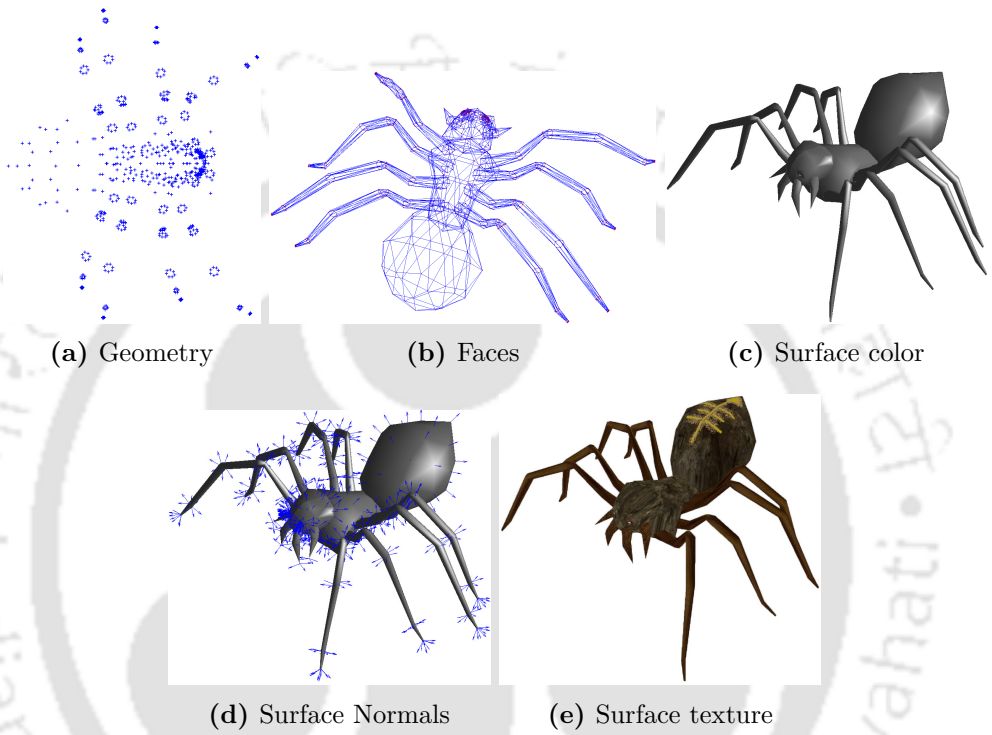
The emerging demand for the use of 3D model data by various applications in networked environments has made researchers to think for efficient representation of such data. The most common representations for the 3D models are using the *3D polygonal meshes*. A polygonal mesh is a finite group of *vertices*, *edges* and *faces* that are used to represent the surface of a 3D model as a solid shape in computer graphics [3]. A vertex signifies a single point in the mesh. An edge defines a line segment connecting two vertices. The number of edges incident to a vertex is called the *degree* (or *valence*) of the vertex. A face is convex polygon in 3D space bounded by edges on the surface of the mesh. A face bounded by three edges is called a triangular face and is an example of a planer polygon. However, a face bounded by more than three edges like quadrilateral, hexagonal or other polygonal forms does not necessarily form a planar polygon. Therefore, the triangular faces are the most popular and widely used for representing the 3D polygonal meshes. They are very effective for storage and easily supported by the graphics rendering hardware of all the manufacturers. This is because:

- A triangle lies always in a plane
- A triangle is convex polygon with all its interior angles less than 180° and
- The interior parameters of a triangle can be easily expressed by the linear combinations of coordinate values of its three vertices.

The different attributes of a triangular mesh are as follows:

- *Geometry* defining the positions of vertices in the 3D space,
- *Topology* or *Connectivity* giving the adjacency information between vertices to form triangular faces covering the mesh surface,

- *Surface colour* defining the colour of the triangular faces and mesh vertices,
- *Surface normal* representing the orientation of triangular faces in the 3D space and
- *Texture* quantifying the variation of the surface roughness or smoothness.



(a) Geometry **(b)** Faces **(c)** Surface color

(d) Surface Normals **(e)** Surface texture

Figure 1.2: Attributes of a 3D triangular mesh ("Spider" model)

The geometry and the connectivity data are the two primary attributes which are needed to view the 3D objects in the wireframe model of triangular meshes as shown in Figure 1.2(b). The other three attributes namely, surface colour, surface normal and surface texture are known as the *property attributes* of the 3D triangular meshes. They are used to enhance the appearance and the details of the wireframe 3D mesh model as shown in Figure 1.2(c), 1.2(d) and 1.2(e) respectively. The property attributes are mostly attached with the vertices in terms of their colours, normal vectors and texture coordinates. The rendering of a 3D mesh with the texture mapping consumes higher graphics memory and increases the rendering time compared to rendering in a wireframe mode without the texture mapping. The wireframe mode helps to visualize the 3D mesh in a faster way by using only the geometry and connectivity data. It helps in quick demonstration on proof of concepts of realistic

(a) Wireframe view **(b)** Enhanced view

Figure 1.3: An example of a static 3D triangular mesh from "Face" model

events using basic 3D models. This thesis work also considers the 3D triangular meshes with the two primary attributes viz., geometry and connectivity information for the compression purpose.

1.2.3 Static vs Dynamic 3D meshes

The 3D triangular meshes can be divided into two categories - *static* and *dynamic* 3D meshes.

Static 3D mesh : A 3D triangular mesh with fixed attributes is called a static 3D mesh. The static mesh represents a 3D model in a single frame. Figure 1.3 shows an example of a static mesh frame of a 3D model, "Face". It consists of two primary information- geometry and connectivity which are invariant of time. The vertex colour and surface colour are assumed to be fixed during plotting. A static 3D mesh can be created by using a 3D modelling software or by transforming a real world object using 3D scanners and triangulation techniques.

Dynamic 3D mesh : A series of static 3D meshes is called a *dynamic 3D mesh* or a *3D animation* or a *3D mesh sequence*. It comprises of consecutive frames of static meshes sampled at different time instants. Each frame in a dynamic mesh sequence will have varying geometry as well as topology or connectivity data over the time duration.

1.2.4 Types of dynamic 3D meshes

Based on the nature of changing attributes per frame of 3D meshes and the motion characteristics, the dynamic 3D meshes are categorised as follows:

a) Rigid body and soft-body 3D animation

(i) In *rigid-body* 3D animation, the distance between any two vertices does not change over time and all the vertices of the triangular meshes move as one entity. The representation of the rigid-body motion is simple but it can not characterize the smooth and realistic motion of animated objects.

(ii) In *soft-body* 3D animation, the relative positions of the vertices are independent of each other. All the vertices can move freely over the animation duration to capture smooth and realistic motions of the 3D objects. Because of the independent movement of each vertex across the frames, it is required to process large data in real time to visualize the soft-body animation. It also restricts the real time transmission of raw data of soft-body animation through usual communication links. This is the major disadvantage of soft-body animation. Some compression techniques need to be employed for the realization of such type of animation.

The main focus of this thesis is to study and develop some compression algorithms for the soft-body 3D animations. The terms 3D animation and 3D mesh sequence are used interchangeably throughout the text to represent the soft-body 3D animations.

b) Isomorphic and non-isomorphic 3D animation

The dynamic 3D meshes are usually created using 3D graphics software from computer simulations of 3D models over a certain time duration. They can alternatively be generated using the virtual reality technique [4, 5] by 3D scanning of the poses at different time instants of real world moving objects followed by triangulations of the each frame. However, there exists a significant difference between the dynamic meshes generated from the above two techniques.

The dynamic meshes generated using 3D graphics modelling software mostly fall under *3D animations* and they are generally created with 3D models having fixed number of vertices and connectivity data. In such a sequence, the instances of a 3D model in successive frames are *isomorphic*, i.e. topologically equivalent, with one-to-one correspondence of the vertices and edges. Therefore, the consecutive frames of 3D animations will have varying geometry in the Cartesian space due to the changing dynamics of the 3D model. Each frame will have fixed number of vertices and constant connectivity data. These 3D mesh sequences satisfy the graph isomorphism property. Figure 1.4 shows an example of an isomorphic 3D animation by plotting a few selective frames from the "Chicken" animation. Each

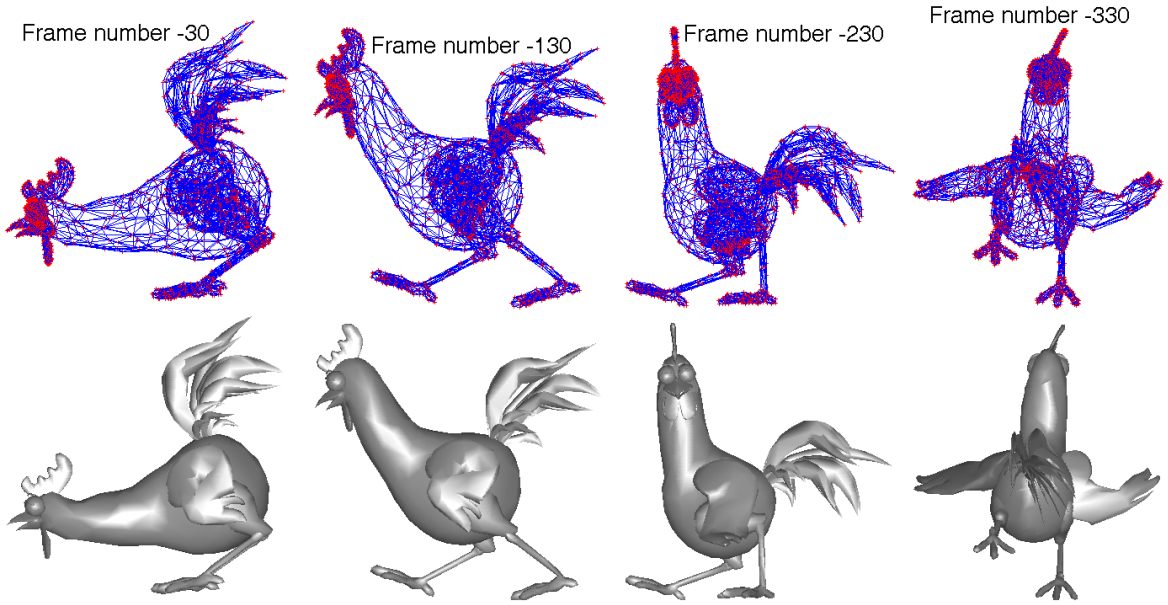


Figure 1.4: Frames of an isomorphic 3D animation with "Chicken" model (3030 vertices and 5664 faces)

frame shown in the figure represents an instant of the moving 3D model in time. The dynamic mesh sequence generated using the virtual reality technique is termed as *time-varying* or *non-isomorphic* 3D animation. Here frames are generated independently from 3D-modelling of the multi-view images of the real-world objects. The instances of the 3D model present in successive frames are not isomorphic to each other. Therefore, the consecutive frames of a time varying 3D animation will have changing geometry as well as connectivity information. Figure 1.5 shows 4 frames of a non-isomorphic 3D animation created with the reconstructed 3D meshes of real world objects at different time instants.

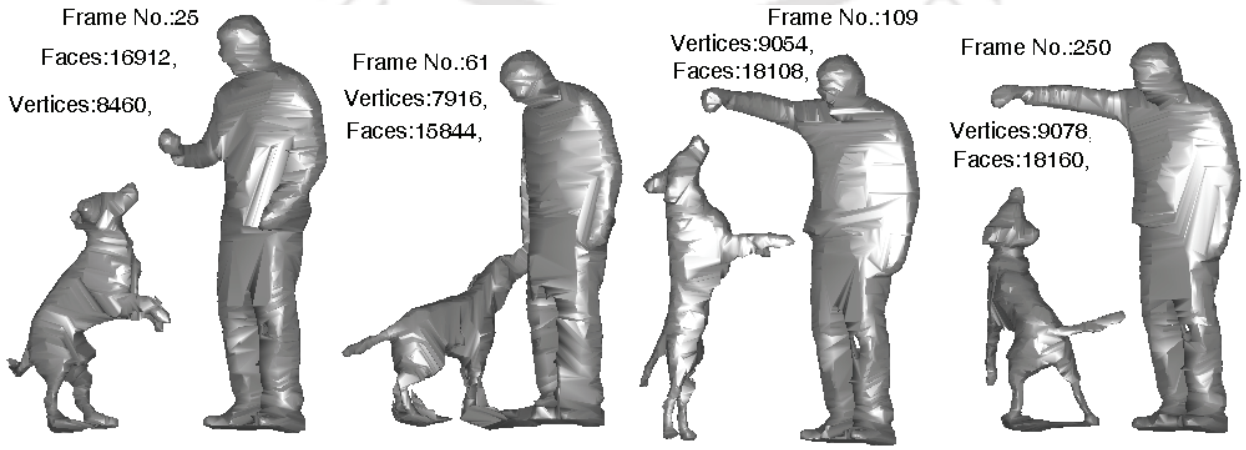
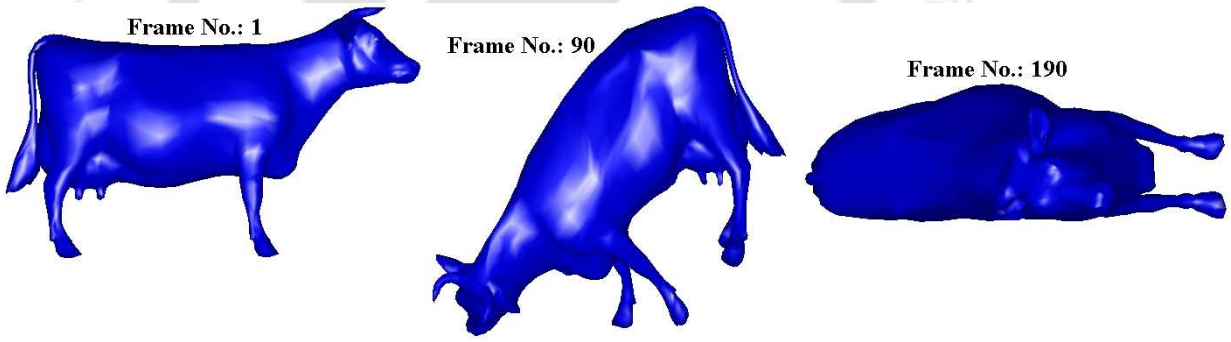


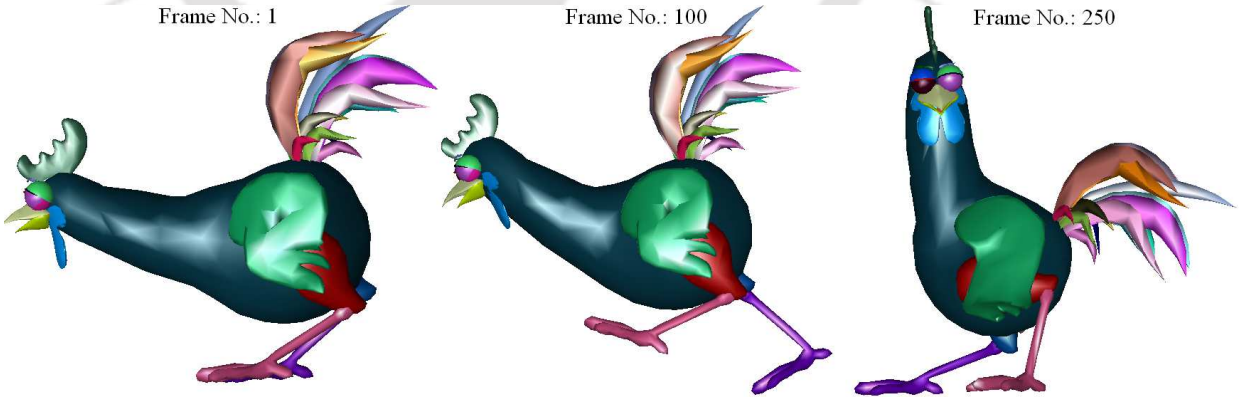
Figure 1.5: Frames of a non-isomorphic 3D animation "Dog-Man-Ball" (http://4drepository.inrialpes.fr/)

An isomorphic 3D animation may contain a single 3D object or multiple 3D objects per frame for visualizing the animation for a specified duration. Accordingly, it may be termed as *isomorphic single object* or *isomorphic multi-object* 3D animation respectively. Similarly, there will be non-isomorphic single-object or multi-object 3D animation for the case of a non-isomorphic 3D animation.

It is observed that the 3D models built using the 3D modelling software may comprise of either single or multiple *connected components* (CCs). These CCs are isolated closed convex polygons which are geometrically close to each other and which represent the 3D surface of different parts of a single 3D model. The individual CCs are isomorphic across all the frames. For example, the single 3D model in the "Cow" animation consists of a single CC, whereas, the single 3D model used in the "Chicken" animation consists of 41 CCs as shown in Figure 1.6(a) and 1.6(b) respectively.



(a) Frames of "Cow" animation with single CC



(b) Frames of "Chicken" animation with multiple CCs

Figure 1.6: Example of 3D animations built using 3D objects with (a) single CC and (b) multiple CCs

c) Block-isomorphic 3D animation

As discussed above, in an isomorphic 3D animation, the number of vertices per frame representing the 3D models is constant and hence the topology relationship among the vertices is also assumed to be fixed over all the frames. However, there are some cases of 3D animations where the number of vertices and connectivity data per frame are constant for certain numbers of frames and then get changed to another set of vertices and connectivity data in between during the animation period. Such a 3D animation is called a *block-isomorphic* 3D animation. Block isomorphism occurs if

- a 3D model gets changed to another 3D model having different number of vertices and connectivity information after certain intervals and
- the number of 3D models present in each frame of the 3D animation gets changed at random intervals during the animation period.

The first category of 3D animation which shows the change in vertex and connectivity involving single 3D model is termed as *block-isomorphic single-object* (BISO) 3D animation. The second category of 3D animations which involves multiple 3D models is termed as *block-isomorphic multi-object* (BIMO) 3D animation. Figure 1.7 and 1.8 show an example of a BISO-3D and a BIMO-3D animation respectively.

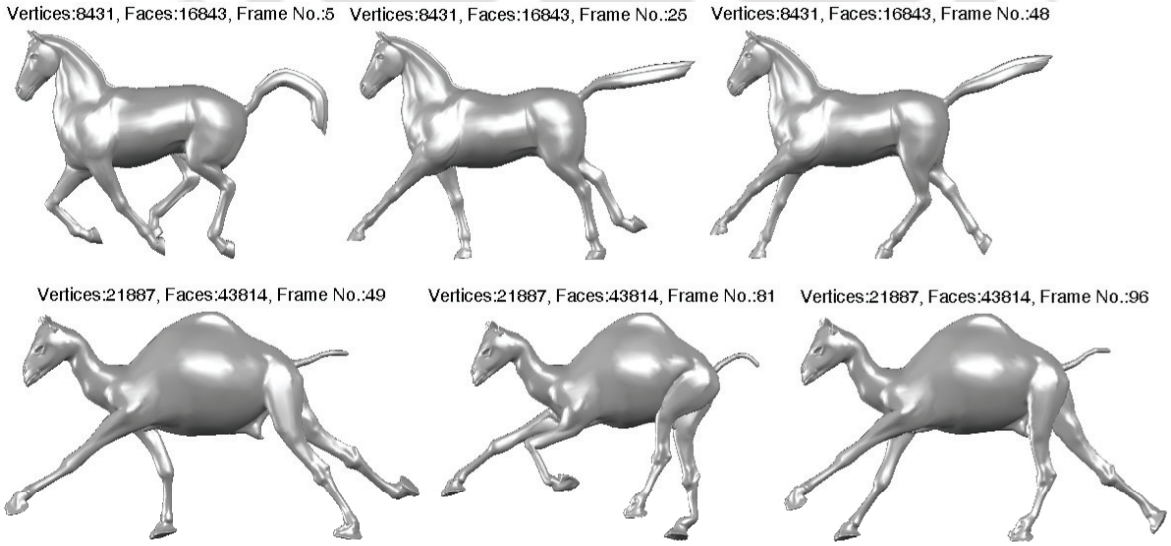


Figure 1.7: Frames of a block-isomorphic single-object 3D animation

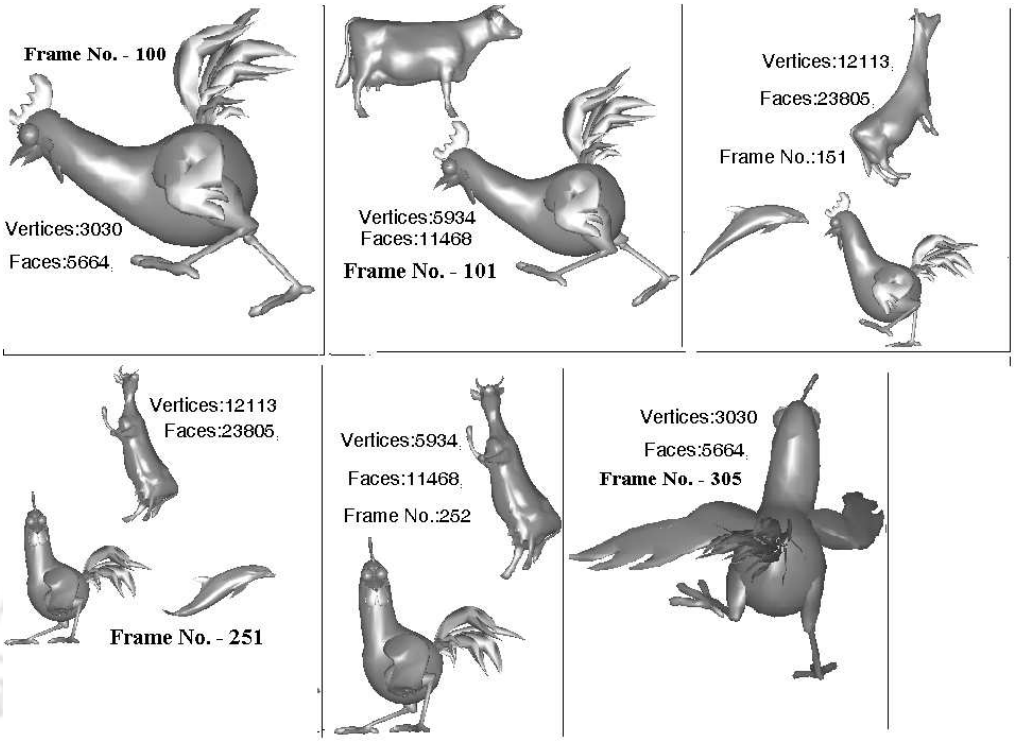


Figure 1.8: An example of a block-isomorphic multi-object 3D animation

In Figure 1.7, a single 3D model "Horse" with 8431 vertices gets changed to 3D "Camel" with 21887 vertices at frame number 49. The frames of the BIMO-3D animation shown in Figure 1.8 comprise of three 3D objects namely, "Chicken", "Cow" and "Dolphin". The number of 3D objects per frame has changed to two, three, two and one at frame numbers 101, 151, 252 and 305 respectively.

In addition to the above types, there exists motion-capture (MoCap) based 3D animations [103] which record human body movements to produce skeletal animations. A MoCap 3D animation system uses sensors placed on the human body to capture the temporal trajectories of the selected parts. It only records changing geometry information of a set of markers or key points over the time without any topology information. They are used for synthesizing human motion to simulate character movements in many interactive applications like entertainment, game development, military, sports, robotics and computer vision. They are not used to design generalized animations using real world 3D objects which need both changing geometry and connectivity data over time.

1.2.5 Mathematical representations of 3D mesh data

The mathematical representations of the geometry and connectivity data for the static 3D mesh and 3D animation are presented below.

1.2.5.1 Static 3D mesh

The geometry and the connectivity information of a static 3D mesh are represented by the model $\mathcal{M}_s = (\mathbf{G}_s, \mathbf{C}_s)$ [6] where $\mathbf{G}_s \in \mathbb{R}^{N_v \times 3}$ and $\mathbf{C}_s \in \mathbb{Z}^{+^{N_t \times 3}}$ denote the geometry and the connectivity data respectively. The connectivity data of the static mesh frame is represented using a set of N_t triangular faces $\mathbf{f}_n$ and is given by

$$\mathbf{C}_s = \{\mathbf{f}_n : \mathbf{f}_n = \{i_n, j_n, k_n\} \in \mathbb{Z}^{+^3}\}\ ,\ \ n = 1, \cdots, N_t, \quad i_n, j_n, k_n \in \{1, \ldots, N_v\} \tag{1.1}$$

The indices i_n, j_n and k_n signify the indices of the adjoining three vertices which are interconnected using the three edges $\{i_n, j_n\}$, $\{j_n, k_n\}$ and $\{k_n, i_n\}$ to form the n-th triangular face $\mathbf{f}_n$. The geometry data $\mathbf{G}_s$ contain the geometry positions of the N_v vertices describing the static 3D mesh and is given by

$$\mathbf{G}_s = \{\mathbf{p}_{v_i} :\ \ \mathbf{p}_{v_i} = (v_{i_x}, v_{i_y}, v_{i_z}) \in \mathbb{R}^3, i = 1, \ldots, N_v\}, \tag{1.2}$$

where $\mathbf{p}_{v_i}$ represents the position of the i-th vertex in 3D space having three coordinate components v_{i_x}, v_{i_y} and v_{i_z} corresponding to X, Y and Z axes respectively.

1.2.5.2 3D animations

An isomorphic 3D animation with constant topology information $\mathbf{C}_d$ and dynamic geometry component $\mathbf{G}_d$ spanning over a time duration using N_f number of frames is mathematically represented as $\mathcal{M}_d = (\mathbf{G}_d, \mathbf{C}_d)$. The topology data $\mathbf{C}_d$ can be defined similarly to the topology data for a static 3D mesh frame as given by Equation (1.1). The geometry data $\mathbf{G}_d$ are defined over a set of N_f frames containing the geometry positions of N_v vertices describing the 3D model in each frame and given by

$$\mathbf{G}_d = \{\mathbf{G}^1, \ldots, \mathbf{G}^f, \ldots, \mathbf{G}^{N_f}\}, \tag{1.3}$$

Here, $\mathbf{G}^f \in \mathbb{R}^{N_v \times 3}$ signifies the geometry of the f-th frame of the animation with a set of N_v vertices per frame and is given by

$$\mathbf{G}^f = \{\mathbf{p}^f_{v_i} :\ \ \mathbf{p}^f_{v_i} = \left(v^f_{i_x}, v^f_{i_y}, v^f_{i_z}\right) \in \mathbb{R}^3, \quad i = 1, \ldots, N_v\}, \tag{1.4}$$

where $\mathbf{p}^f_{v_i}$ represents the position of i-th vertex of the frame f in the 3D space having three coordinate components $v^f_{i_x}$, $v^f_{i_y}$ and $v^f_{i_z}$ corresponding to X, Y and Z axes respectively.

The overall geometry component of the 3D animation defined by the set $\mathbf{G}_d$ as shown in Equation (1.3) is also represented in a 2D matrix form for the easy application of signal processing algorithms on it. Two different types of matrix representations for the overall animation geometry are made - (a) *frame-based* representation and (b) *trajectory-based* representation.

a) In the *frame-based* representation, the geometry of each frame $\mathbf{G}^f$ is stored as a column vector $\mathbf{a}_f$ of length $3N_v$ by cascading the three coordinates, v_{ix}^f, v_{iy}^f and v_{iz}^f of the N_v vertices of the frame and given by

$$\mathbf{a}_f = \left[v_{1_x}^f \ \ldots \ v_{N_{v_x}}^f v_{1_y}^f \ldots v_{N_{v_y}}^f v_{1_z}^f \ldots v_{N_{v_z}}^f \right]^T, \quad f = 1, \ldots, N_f \tag{1.5}$$

Accordingly, the whole animation geometry for N_f number of frames can be represented by a matrix $\mathbf{A}$ of dimensions $3N_v \times N_f$ [7] as

$$\mathbf{A} = \left[\mathbf{a}_1 \mid \mathbf{a}_2 \mid \ldots \mid \mathbf{a}_{N_f} \right] = \begin{bmatrix} v_{1_x}^1 & v_{1_x}^2 & \ldots & v_{1_x}^{N_f} \\ \vdots & \vdots & \ldots & \vdots \\ v_{N_{v_x}}^1 & v_{N_{v_x}}^2 & \ldots & v_{N_{v_x}}^{N_f} \\ v_{1_y}^1 & v_{1_y}^2 & \ldots & v_{1_y}^{N_f} \\ \vdots & \vdots & \ldots & \vdots \\ v_{N_{v_y}}^1 & v_{N_{v_y}}^2 & \ldots & v_{N_{v_y}}^{N_f} \\ v_{1_z}^1 & v_{1_z}^2 & \ldots & v_{1_z}^{N_f} \\ \vdots & \vdots & \ldots & \vdots \\ v_{N_{v_z}}^1 & v_{N_{v_z}}^2 & \ldots & v_{N_{v_z}}^{N_f} \end{bmatrix}_{3N_v \times N_f} \tag{1.6}$$

b) In the *trajectory-based* matrix representation [8] [9] of the animation geometry, the frame-wise geometrical positions of each vertex $\mathbf{p}_{v_i}^f$ across the N_f number of frames are cascaded together to form a trajectory vector $\mathbf{t_i}$ of length $3N_f$ given by

$$\mathbf{t_i} = \left[v_{i_x}^1 \ v_{i_y}^1 \ v_{i_z}^1 \ v_{i_x}^2 \ v_{i_y}^2 \ v_{i_z}^2 \ \ldots \ v_{i_x}^{N_f} \ v_{i_y}^{N_f} \ v_{i_z}^{N_f} \right]^T, \quad i = 1, \ldots, N_v \tag{1.7}$$

The trajectory vectors $\mathbf{t_i}$ for N_v number of vertices are placed as the rows of a matrix $\mathbf{B}$ of

dimension $N_v \times 3N_f$ as

$$\mathbf{B} = \begin{bmatrix} \mathbf{t}_1 \\ \mathbf{t}_2 \\ \vdots \\ \mathbf{t}_{N_v} \end{bmatrix}^T = \begin{bmatrix} v_{1_x}^1 & v_{1_y}^1 & v_{1_z}^1 & v_{1_x}^2 & v_{1_y}^2 & v_{1_z}^2 & \cdots & v_{1_x}^{N_f} & v_{1_y}^{N_f} & v_{1_z}^{N_f} \\ v_{2_x}^1 & v_{2_y}^1 & v_{2_z}^1 & v_{2_x}^2 & v_{2_y}^2 & v_{2_z}^2 & \cdots & v_{2_x}^{N_f} & v_{2_y}^{N_f} & v_{2_z}^{N_f} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ v_{N_{v_x}}^1 & v_{N_{v_y}}^1 & v_{N_{v_z}}^1 & v_{N_{v_x}}^2 & v_{N_{v_x}}^2 & v_{N_{v_x}}^2 & \cdots & v_{N_{v_x}}^{N_f} & v_{N_{v_y}}^{N_f} & v_{N_{v_z}}^{N_f} \end{bmatrix} \tag{1.8}$$

The main objective of animation geometry compression algorithms is to represent the matrix **A** and **B** in a more compact manner by applying the data compression algorithms.

1.3 Compression of 3D meshes

1.3.1 Need for compression

The goal of the 3D mesh compression algorithms in the early 90s was to reduce the amount of bytes required to be transferred from the computer main memory to the graphics card memory for the fast rendering purpose. With that motive, a class of algorithms were proposed to compress both the triangular and the polygonal meshes. But the goal of present compression algorithms is not merely limited to the benefits of hardware processing. There are other compression goals which need to be addressed. They are discussed below:

(i) **Efficient storage, transmission and rendering for 3D static mesh**

The sizes of 3D models are now-a-days becoming larger due to the surface reconstruction of huge point clouds generated from the modern scanning devices. The generation of the complex 3D models is also possible with the advent of new 3D modelling software. The large size of static 3D models is because of the large number of vertices and polygonal faces for visualizing the minute details of the 3D models. Eventually, the large size will increase the cost of storage and the processing time by the graphics hardware. Furthermore, the transmission of these large 3D models over the limited bandwidth internet connection for real time visualization in another challenge. It becomes utmost necessary to develop efficient compression algorithms for the huge 3D static meshes for decreasing the cost of storage hardware, enhancing rendering performance and enabling fast transmission over the network.

(ii) **Efficient storage and broadcasting of 3D animation**

With the advancement of multimedia technologies, the popularity of 3D animations featuring 3D objects has increased. For a realistic visual experience, the 3D animations are made with complex static 3D meshes spanning over hundreds of frames to describe a short sequence lasting for a few seconds. Therefore, a dynamic 3D mesh requires more storage space as compared to a static 3D mesh. Moreover, the raw animation data with complex 3D models always require more reconstruction time for real-time visualization over the internet. So, the development of a good compression algorithm for the efficient storage and broadcasting of 3D animations while maintaining a good visual quality has become more challenging. This area is gaining much interest among researchers in the present scenario. We have also focused our work for the development of compression algorithms for 3D animations only.

(iii) **Scalability**

Now-a-days, different users have different hardware devices like laptops, desktop PCs and mobile phones for visualizing the same animation content with different resolutions. Moreover, the users may also access the animated 3D models over heterogeneous networks with different bandwidths like Internet, LAN, wireless networks etc. Thus, there is a growing demand to compress the animation data in a scalable manner. The scalable algorithms facilitate the encoding the animation data only once using a single bitstream having different sub-layers and then decoding the bitstream with sub-layers in multiple ways in terms of frame rates, resolutions or visual qualities. Like in image and video coding, the most common types of scalability used in 3D animations are spatial, temporal and quality scalability. In spatial or temporal scalability, the decoder decodes the subset of encoded bitstream with variable spatial or temporal resolutions respectively, whereas in quality scalability, the bitstream is decoded with variable distortions. The development of efficient compression algorithms supporting spatial, temporal and quality scalability is an active research topic among the researchers in recent times.

1.3.2 Redundancies in 3D animation data

The goal of any compression algorithm is to represent the raw input in a compact way by removing the irrelevant information and reducing the redundant information. The 3D meshes acquired by the 3D

acquisition hardware also contain irrelevant and redundant information. The irrelevant information mainly signifies the noise or outlier samples which may be introduced during the acquisition process or any other information not relevant to human perceptual point of view. For example, a 3D mesh model after acquisition from a 3D laser scanner may be represented using the 32 bits per coordinate component of a vertex. However, during displaying the 3D model on a PC monitor, the same vertex points are represented in 16 bits per coordinate component and it may be perceived equally well as the 32-bit representation by human observer provided it is not observed by zooming very close to the model's surface to view the noticeable distortions. Although certain distortions will be introduced by this process, the removal of such irrelevant data will definitely reduce the bitrate to half. So, the required distortion i.e. the inverse of quality is a function of the bitrate. This trade-off between the target distortion vs. minimum bitrate and vice versa can be analysed theoretically using the *rate-distortion* (RD) theory [10].

The reduction of redundancy in data exploits the statistical dependencies present among the sample data. In a 3D animation, there persists redundant information among the vertices in the spatial as well as in the temporal direction. To remove the redundancy, prediction methods are mostly used in the spatial and temporal domains. For instance, a spatial group of neighbouring vertices representing some parts of the 3D object surface may exhibit similar motions across the animation frames. Instead of encoding each vertex of the group separately, the prediction method can be applied to predict the actual position of a vertex in a frame using the already encoded positions of its nearest neighbourhood vertices in the past and present frames. The difference in the predicted value and the actual value of the vertex i.e. the residue will be of a small dynamic range and have reduced variance. The residues can be efficiently encoded with fewer bits compared to the original position using the entropy coding. The entropy encoder generally uses the statistics of the of the source represented by the probability density function (PDF) to apply the necessary variable length coding (VLC) algorithms like the Huffman coding [11], the arithmetic coding [12] and the context adaptive binary arithmetic coding (CABAC) [13] to get the compact bitstream.

Another mechanism followed to remove redundant and irrelevant data present in 3D animation is by the application of some linear transformation techniques on the original geometry data to get a set of uncorrelated data. This helps in better processing of the data by the compression algorithms

in the transform domain. The transformation operations are invertible in most of the cases and so the original data can be reconstructed back from the samples in the transform domain. Some of the important transformation methods that find use in the 3D animation geometry compression are the discrete cosine transform (DCT) [14], the discrete wavelet transform (DWT) [15], the principal component analysis (PCA) [7], the singular value decomposition (SVD) [16] and the spectral transform using the graph Laplacian matrix [17] and so on. The PCA [104] and the DWT [105], [103] have also been used to exploit the spatio temporal redundancies to compress the MoCap data. It is also observed that the transformation operations like the DWT and the SVD can produce scalable or progressive representations of the original animation data. This feature has great importance for the applications related to real time visualization of the animation data over the Internet. The scalable or progressive coding of the original data helps in partial decoding of the bitstream to get a low quality reconstructed animation without waiting for the time to download the whole encoded bitstream. The subsequent decoding of more data gradually improves the quality.

1.3.3 Types of compression for 3D meshes

The compression of 3D polygonal meshes involves three types of compressions, i.e., connectivity compression, geometry compression and geometry property data compression. Out of these, the compression of connectivity and geometry data is primarily important for the compact representation of 3D meshes. The compression algorithms need to compress both the geometry and connectivity data separately. The connectivity data are generally encoded in a lossless manner as they are represented in discrete integers describing the triangular faces. The geometry data which are represented in 32-bit floating point values are mostly encoded in a lossy manner by representing them in fewer number of bits. The compression algorithms for 3D meshes can be grouped in two broad categories: a) *single-rate* or single resolution and b) *progressive* or multi-resolution compression algorithms.

(a) Single-rate compression

The single-rate or single resolution compression methods allow the compact representation of the input mesh using a single bit-stream containing both geometry and connectivity data. The decoder can reconstruct the input mesh for the visualization by the end user only if the full compressed bitstream is available. It does not allow any intermediate reconstruction of mesh data from the compressed

bitstream. This type of compression is important for the compact storage and the fast transmission of the compressed 3D mesh data over the Internet.

(b) Progressive compression

The progressive compression algorithm [18–21] on the other hand allows successive reconstruction of intermediate 3D meshes with different level of details (LoD) at the decoder as more compressed bitstream is received. This type of compression method enables an end user to have an early preview of the coarse version of the input 3D mesh without downloading the full compressed bitstream. The end user can select the required LoDs as per his satisfaction based on the hardware capability and network constraints. However, there exists a rate-distortion trade off between the compressed bits and the LoDs in progressive compression. This type of compression is suitable for the transmission of 3D meshes over limited bandwidth network. The progressive compression methods can not provide better coding gain compared to a single-rate coder as they cannot fully exploit the redundancy among the mesh data as exploited by the single-rate coders.

The progressive mesh compression methods are either geometry or connectivity driven. In the geometry driven progressive mesh compression schemes [17, 22–24], the geometry data is given due priority to compress them in a progressive manner with different spatial geometry based layers. The compression of connectivity data is driven by the geometry coding. The wavelet based multi-resolution mesh coding methods [15, 25–27] are generally belong to the geometry driven progressive mesh coding. In the connectivity driven progressive mesh compression techniques [28–30], the compression of connectivity data in progressive manner with different LoDs is given due priority. The geometry compression is driven by the connectivity coding. This class of algorithms work in the principle of mesh simplification based on connectivity information to give required LoDs. The connectivity data is gradually simplified by reducing the number of edges, vertices or faces stage by stage at each LoDs.

1.3.4 Scope for compression of 3D animations

The raw data size of 3D animations require a large storage size because of larger number of static frames and so it is necessary to compress the 3D animations in an efficient manner for compact representation. The compression of raw data can be achieved by exploiting the spatio-temporal coherence of geometry data in the successive frames. Based on coherence of connectivity information across the frames, the compression methods of 3D animations can be divided into two categories.

The first category is for compressing animations with constant connectivity i.e.isomorphic 3D animations and the second category is for animations with varying connectivity i.e. non-isomorphic 3D animations. Most of the existing geometry compression methods reported in literature deal with the compression of isomorphic 3D animations comprising a single object per frame across the full animation frames. The existing compression methods for isomorphic 3D animations can be classified into five categories: *segmentation-based* methods [8, 31–33], *PCA-based* methods [3, 7, 9, 34], methods *based on spatio-temporal prediction* [35–39], *DWT-based* methods [27, 40, 41], and finally the *MPEG based* [2, 42] animation compression algorithms. Many of the existing methods for animation geometry compression use the PCA as the important tool for single rate compression. The PCA approaches are efficient because it analyses the coherence of the entire sequence and give best compression results when processed off-line on the 3D animation data. Compared to the direct PCA, the PCA on clusters provides better compression of animation geometry by exploiting both the spatial and temporal coherence across the sequences. The improvements for PCA based geometry compression methods are reported in clustered PCA (CPCA) [8] and the local PCA (LPCA) [43] methods.

In the case of non-isomorphic 3D animations, the complexity of compression methods is more compared to the isomorphic case. This is because of non-coherence of spatio-temporal information across the frames. It is much harder to exploit the spatio-temporal redundancies in the case of non-isomorphic 3D animations without prior mapping of geometry and connectivity information across the frames. Two compression methods proposed by Han et al. [44] and Yamasaki and Aizawa [45] deal with non-isomorphic 3D animations. These methods apply blocking matching techniques used in video compression methods. They divide the 3D meshes in each frame into number of blocks or patches and perform matching of those blocks/patches in subsequent frames and finally code the residuals.

Moreover, in real animations, the geometry as well as the connectivity of vertices in the 3D animations may change at irregular intervals over time span. In that case, the animation data may satisfy isomorphic property in blocks of frames. No compression method can be found in literature for the block-isomorphic 3D animations. The research work presented in this thesis centres around the development of efficient algorithms for the compression of isomorphic and block-isomorphic animation geometry data. The scalable compression of animation data is another emerging issue that is evolving now-a-days because of the requirement of visualizing the 3D animation at different dimensions and

rates as discussed below.

1.3.5 Scalable coding of 3D animation

Multimedia data can be encoded into two different ways: *non-scalable coding* and *scalable coding*. In non-scalable coding, the data is encoded and decoded independent of actual channel characteristics. Non-scalable coding gives high quality and a simple decoder, but it requires a large storage size. The main problem with non-scalable coding is that, it is difficult to adapt the bit stream over different needs of the users and different channel bandwidths. On the other hand, with scalable coding, multimedia data needs to compress in a single scale and depending on various needs of the users and network or bandwidth condition, the bit stream is decompressed in a multiple scale with different qualities, spatial resolutions and/or temporal resolutions. The compressed bitstream is encoded into a *base layer* and a few *enhancement layers*, where the enhancement layers add spatial, temporal, and/or quality scalability to the reconstructed base layer. Scalable coding gives small storage size, simple bit-stream switching and multi-cast applications with complicated decoder and less compression efficiency compared to non-scalable coding.

1.3.5.1 Types of scalability

There are three types of scalability: temporal, spatial, and quality or SNR scalability.

(i) Temporal scalability

The sub-stream of a bit-stream having temporal scalability provides a lower frame rate or temporal resolution than the complete bit-steam of the multimedia data. It can be achieved using the group of frame (GOF) structure. The GOF can be defined as the group of all the frames which are temporally located between two consecutive key frames including the next key frame. Let, the temporal layer can be represented by the temporal layer identifier T. Thus T_0 represents base layer, T_1 represents the enhancement layer 1, and so on. The concept of hierarchical B-frames is used to achieve temporal scalability with dyadic temporal enhancement layers as illustrated in Figure 1.9. The numbers written below the frames indicate the coding order.

(ii) Spatial scalability The sub-stream of a bit-stream having spatial scalability provides a smaller spatial resolution than the complete bit-steam of the multimedia data. It can be achieved by

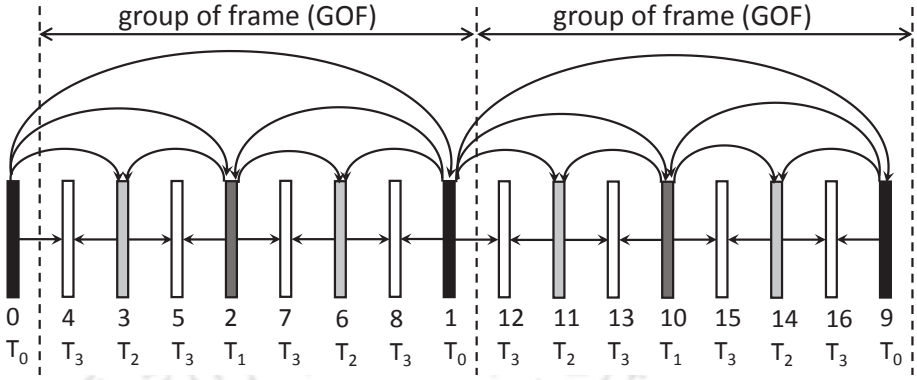


Figure 1.9: Prediction with hierarchical B-frames for temporal scalability [1]

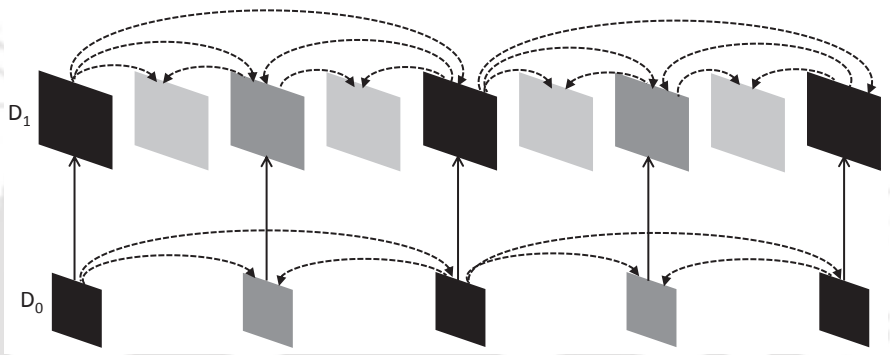


Figure 1.10: Multi-layer structure with inter-layer prediction for spatial scalability [1]

using a multi-layer coding approach. Each layer has a different spatial resolution and can be represented by the spatial layer identifier D. Thus D_0 represents base layer, D_1 represents the enhancement layer 1, and so on. Each spatial layer can be encoded using motion compensation and intra-prediction as a single-layer coding. The coding efficiency can be improved by adding inter-layer prediction mechanisms as shown in Figure 1.10.

(iii) Quality scalability The sub-streams of the bit-stream having quality scalability provide the same resolution or the frame rate (spatio-temporal resolution) as the complete bit-steam of the multimedia data. But They have different qualities or signal-to-noise ratios (SNR). Quality scalability is also called as SNR scalability. In quality scalability, the spatial resolution of the base and enhancement layers are same. So it is referred as a special case of the spatial scalability. Quality scalability can be achieved by repeatedly decreasing the quantization step size and encoding the difference in quantized coefficients.

The scalability concepts in the SVC are to be appropriately applied for animation geometry compression. It will be useful for low-latency streaming applications of 3D animations especially for handheld devices over limited bandwidth channels to view the 3D animation at a reduced number of mesh vertices (spatial scalability) or reduced frame rates (temporal scalability) or less visual quality (quality scalability). Unlike video, the scalability issues of 3D animation geometry compression is not well-addressed. There are a few approaches that present the scalable compression of 3D animations [25, 33, 42, 46]. Most of these methods have not reported the scalability performance. It is observed that the scalable compression methods for animation geometry has its own challenges, particularly for the spatial scalability case. Thus, there is scope for further research.

1.3.6 TFAN connectivity compression

Unlike the other multimedia data such as speech, image and video, the compression of geometry data of 3D meshes depends on the underlying structure of the mesh data. The connectivity data define the underlying structure of a 3D mesh based on which the geometry data can be compressed at the encoder side. Similarly, the decoder must know this structure in advance before reconstructing the geometry data. So, it is necessary to compress the connectivity data efficiently in lossless manner along with the 3D animation geometry. In general, the compression of connectivity data involve traversing the mesh from vertex to vertex to detect the occurrence of a few predefined patterns in the mesh. The detection of the patterns are recorded as a sequence of different symbols. The list of generated symbols are finally entropy coded to get the compressed data. Some of the popular connectivity compression techniques available in literature of 3D triangular meshes are the Edgebreaker algorithm of Rossignac [30], the valence-based coder by Touma and Gotsman [29] and the Triangle FAN-based (TFAN) compression approach by Mamou et al. [47].

In this work, we have applied the TFAN encoder for the compression of connectivity data of the 3D animation. A TFAN is defined a set of T connected triangles with $T+2$ vertices with a common central vertex. The TFAN algorithm starts with decomposing the mesh into a set of TFANs. The vertices in the mesh are traversed from neighbour to neighbour to detect the TFANs. A triangle in a TFAN is marked 'visited' if it is a part of an earlier TFAN. Based on the 'visited' and 'non-visited' triangles, 10 distinctive TFAN configurations are identified. The indices for the TFAN identified during traversal

of vertices of the mesh are efficiently encoded based on their relative frequencies of occurrences in the mesh. The TFAN codec is shown to outperform the existing MPEG-4/3DMC (3D Mesh Coding) [48] and Touma and Gotsman [29] approaches with better compression efficiency and lower decoding time. The details about the TFAN compression algorithm is available in [47].

1.4 Distortion metrics for geometry compression algorithms

The geometry data of 3D animations undergo various processing operations during compression methods. Geometric distortions are introduced in the original data because of these processing operations. The geometrical distortions result in visual artefacts in the reconstructed animation data and alter the perceptual or visual quality of the 3D content with respect to the original one. Hence, the quality assessment of 3D animations using a *distortion metric* (DM) is an important issue for the development and optimization of the geometry compression algorithms. The quality assessment of dynamic 3D meshes involves the subjective or objective evaluation of the reconstructed 3D mesh data with respect to its original version based on the detectability of artifacts. Since subjective evaluation is a time consuming process, objective DMs are used for faster evaluation of quality of reconstructed 3D meshes. The objective DMs should give outputs correlating with the subjective evaluation scores of distorted animations. For that, it is required to integrate the perceptual characteristics of *human visual system* (HVS) like curvature information, surface roughness, edge lengths, dihedral angles, texture mapping and other psycho-perceptual factors. Accordingly, the existing objective DMs can be divided into *non-perceptual* and *perceptual* types based on their nature of integration of properties of HVS in them. The non-perceptual DMs are computed simply based on the geometrical distances of the vertices in the original and the corresponding reconstructed animation. The perceptual DMs are computed by integrating the characteristics of human visual system (HVS) and show better correlation with the subjective evaluation scores. In this thesis, we have used the following DMs for performance evaluation of geometry compression algorithms.

- *KG error (KG_{err})*: The KG_{err} metric, proposed by Karni and Gotsman [3], is a non-perceptual objective DM widely used by the researchers in for measuring performance of the geometry

compression algorithms for 3D animations and it is defined as

$$KG_{err} = \frac{\left\|\mathbf{A} - \tilde{\mathbf{A}}\right\|_f}{\|\mathbf{A} - \mathbf{E}(\mathbf{A})\|_f} \times 100\% \tag{1.9}$$

where $\mathbf{A}$ is the original geometry matrix given by Equation (1.6), $\tilde{\mathbf{A}}$ is the reconstructed geometry matrix and $\mathbf{E}(\mathbf{A})$ is the mean matrix of size $3N_v \times N_f$ with each column consisting of the mean frame of the matrix $\mathbf{A}$ over the N_f columns. The notation $\|.\|_f$ represents the "Frobenius norm" or "entry-wise Euclidean norm" given by the square root of the sum of the squares of the elements of the matrix.

- *Spatio-temporal edge difference (STED)*: The *STED* metric, proposed by Vasa and Skala [49], is the first perceptual based DM designed for quality assessment of distorted 3D animations. It is based on relative changes in local edge lengths instead of global changes of vertex positions between two mesh sequences. It is defined as

$$STED(d, w) = \sqrt{STED_s(d)^2 + c^2.STED_t(w)^2} \tag{1.10}$$

where $STED_s(d)$ signifies the spatial error, $STED_t(w)$ signifies temporal error and c is a weighting parameter. The spatial error corresponds to the sum of standard deviation of relative edge lengths computed at each vertex within its topological neighbourhood for all the frames. The temporal error is computed as the sum of relative changes of virtual temporal edges between the two mesh sequences. A virtual temporal edge corresponds to the edge formed by connecting the position of a vertex in two consecutive frames. The details of the *STED* metric are presented in Section 5.3.2.

The two metrics used for the measurement of compression performance are the *compression ratio* (CR) and the compression rate expressed in terms of *bits per vertex per frame* ($bpvf$) [8] as defined below.

- *Compression ratio* (CR): The compression ratio for the geometry data is defined as the ratio of total number of elements in the original animation matrix to the total number of elements in

the compressed data. It is given as

$$CR = \frac{\texttt{Total size of original geometry data}}{\texttt{Total size of compressed data}} \tag{1.11}$$

- *Bits per vertex per frame* ($bpvf$): It signifies the compression rates in terms of number of bits per vertex per frame. It is a unit for bandwidth usage measurements. It is defined as

$$bpvf = \frac{\texttt{Total size of encoded bits}}{N_v \times N_f} \tag{1.12}$$

where N_v is the number of vertices per frame and N_f is the number of frames in the 3D animation.

1.5 Motivation and problem formulation

Based on the literature survey, four research issues in animation geometry compression are identified. Accordingly, four research problems are formulated.

1.5.1 Compression of isomorphic animation geometry

In an isomorphic 3D animation, the consecutive frames will have only varying geometry in the Cartesian space with fixed number of vertices and constant connectivity data [50]. In such animations, the instances of the 3D models in successive frames are isomorphic, i.e. topologically equivalent, with one-to-one correspondence of the vertices and edges. There is an increasing demand for designing of complex 3D models to have realistic visual experience in computer generated 3D movies, animations, scientific visualization of 3D models, 3D imaging and 3D video games. These require large storage space for the raw animation data. So, there is a growing demand to develop different compression techniques for representing the raw animation data compactly while maintaining a good visual quality. The compact representation should also facilitate reliable transmission of mesh data through the band limited channel. Thus, optimized compression of 3D animation geometry data is of interest among researchers. This leads us to develop an efficient compression algorithm for an isomorphic 3D animation.

1.5.2 Scalable compression of isomorphic animation geometry

There is a growing need to develop algorithms that can encode the animation geometry data only once and decode them in multiple ways in terms of frame rates, resolutions or qualities. This makes

the scalability as an important issue in the compression of animation data. A scalable geometry compression algorithm encodes the time varying geometry data in a single bit-stream with different sub-layers and decodes the bit-stream in layers to view the full animation in multiple scales. The non-uniform connectivity of the vertices brings additional challenges to compress the animation data in a scalable manner.

1.5.3 Compression of multi-object based animation geometry

In real world animations there may multiple 3D objects in each frame and the geometry as well as the connectivity of vertices of constituted multiple 3D objects may be changing at irregular intervals over time span. Most of the compression works reported in literature deal with 3D animations featuring a single object. No work has been reported in the literature for the compression of animation geometry data having multiple 3D objects. The multi-object based 3D animations contain multiple complex 3D models with changing attributes like vertices, connectivity, face normals and texture across the frames. The size of raw data for a multi 3D object based animation will be much higher compared to the single 3D object based animation. So, there is a need to address the compression issues for such multi-object based 3D animations.

1.5.4 Distortion metrics for animation geometry compression

To evaluate the performance of 3D animation geometry compression algorithms, it is necessary to do the quality assessment of the decompressed mesh data. The quality assessment of 3D animations involves the subjective or objective evaluation of the reconstructed 3D animation data with respect to its original version based on the detectability of artifacts. The subjective evaluation refers to the quality assessment of the reconstructed 3D animations by human observers in terms of opinion scores based on the perceptual quality. However, carrying out subjective assessment is always time consuming and cannot be integrated as an automatic process. The researchers have, therefore, focused on the development of objective distortion metric (DM) which can numerically predict the quality. This metric should have correlation with the subjective visual quality score. Thus, it is required to develop the DM to predict the quality of any distorted 3D animation in an automatic manner correlating with the subjective mean opinion score by integrating the properties of the human visual system (HVS).

1.6 Thesis contributions

The objective of this thesis is to compress the geometry data of isomorphic and block-isomorphic 3D animations for efficient storage and transmission over the heterogeneous network. It also investigates the approaches to compress the geometry data in a scalable way, adapting different resolutions, qualities and transmission rates. This thesis reports four broad contributions in the area of 3D animation geometry compressions including a distortion metric as given below.

1) Improving the performance of the CPCA method

The first contribution is related to the non-scalable compression of isomorphic 3D animation geometry. A method has been proposed for improving the performance of an existing geometry compression method based on clustered principal component analysis (CPCA). In the CPCA algorithm, the vertex trajectories are clustered using the K-means algorithm followed by the principal component analysis (PCA) on the clusters. However, the compression performance of the existing CPCA method is restricted to the initial random selection of the cluster centres. The proposed method uses the subtractive clustering technique to find stable cluster centre trajectories based on density functions. The stable centres are used as initial cluster centres for the K-means algorithm to cluster the vertex trajectories. The PCA is applied on individual clusters for data compression. A non-uniform quantization method is applied to encode the elements of PCA eigen vectors, PCA coefficients and the mean vertex trajectory of each cluster. A block-based arithmetic coding is proposed to encode the groups of quantization levels obtained using same quantization bit in each cluster. The simulation results on standard 3D animation sequences show stable and better performances than CPCA and LPCA algorithms in terms of objective distortion metric.

2) Scalable compression of animation geometry using singular value decomposition

The second contribution is related to the scalable geometry compression of isomorphic 3D animations. The scalable compression facilitates compression of the geometry data in a single scale and decompressing it in multiple scales. An encoder and decoder structure have been proposed to obtain spatio-temporal scalability using the singular value decomposition (SVD). After the SVD, the elements of the spatial singular vectors are first arranged into different spatial layers starting from a base layer

to enhancement layers. The spatial prediction errors between the elements of successive spatial layers are computed and subsequently quantized non-uniformly and entropy coded. Similarly, the elements of temporal singular vectors are grouped frame-wise from base layer to enhancement layers to compute the temporal prediction errors. The temporal prediction errors are also quantized non-uniformly and entropy encoded. The performance of the proposed SVD based spatio-temporally scalable algorithm is found better than existing scalable compression methods.

3) Compression of block-isomorphic multi-object animation geometry

The third contribution is related to a novel compression scheme for 3D animations featuring multiple 3D objects per frame. An object-based PCA (OPCA) method has been proposed to compress the geometry data of block-isomorphic multi-object (BIMO) 3D animations. This compression method exploits the spatio-temporal correlations among the vertices of the individual 3D objects across the animation length. The animation geometry data of consecutive frames are first segmented into different isomorphic multi-object (IMO) blocks based on the information of changing vertices and their connectivity data. The vertices belonging to individual 3D objects in each IMO block are then detected using a proposed connectivity bases spatial segmentation algorithm. The geometry data of the individual 3D objects are then separated out for each IMO block. The 3D objects spanning across multiple IMO blocks are detected as unique 3D objects based on the mesh attributes and the Euclidean distances between the objects in consecutive IMO blocks. The geometry data of the unique 3D objects are appended temporally across the frames in IMO blocks. The OPCA algorithm is finally applied on the vertex trajectories of the unique 3D objects to get the compression. The compression performance of the proposed OPCA method is evaluated on a few BIMO-3D animations. The simulation results show that at lower bitrate, the proposed OPCA method gives better results by providing minimum perceivable distortions compared to the PCA and the CPCA method. The performance has also been compared with the segment adaptive PCA (SA-PCA) and segment adaptive OPCA (SA-OPCA) methods which apply the PCA and OPCA algorithm respectively on geometry data of each IMO block without any merging operation. The compression performance of the merged OPCA is found better than the SA-PCA and the SA-OPCA methods.

4) An improved distortion metric for 3D animation geometry

The fourth contribution is related to the proposed perceptual visual distortion (PVD) metric for quality assessment of 3D animation geometry compression algorithms. The work aims at discussing the issues with the existing PVD metrics, the $STED$ and the dynamic mesh perceptual distance ($DMPD$), used for quality assessment of 3D animations and proposes an improve PVD metric named as *affine invariant spatio temporal edge difference* ($AISTED$). This new metric incorporates a few properties in the context of correlation with the HVS. It uses the relative changes in spatial and temporal edge lengths between the original and the reconstructed animations to measure the distortions in perceptual manner. The proposed metric is made invariant to any combinations of frame-base affine transformation operations, in terms of translation, rotation and scaling, applied on the animations.

1.7 Organisation of the thesis

The rest of the thesis is organised as follows:

Chapter 2 discusses the existing compression algorithms of 3D animation geometry based on the PCA and the clustered PCA (CPCA) algorithms. It presents a method of improving the performance of CPCA method using the proposed subtractive clustering based CPCA (S-CPCA) algorithm. In **Chapter 3**, scalable compression methods of 3D animation geometry are discussed. It presents a proposed spatio-temporally scalable compression of animation geometry using the SVD. In **Chapter 4**, the animation geometry compression method for multi-object 3D animations is presented. It discusses the proposed object-based PCA (OPCA) method for geometry compression of IMO-3D and BIMO-3D animations. **Chapter 5**, presents an overview of the different distortion metrics (DMs) that are usually used for the performance evaluation of 3D mesh compression algorithms. An improved perceptual DM has been proposed for 3D animation to overcome the limitations of the existing DMs. In **Chapter 6**, the conclusions are drawn with a summary of the major contributions from this thesis work and future scopes for research are presented.

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati

2

Animation Geometry Compression using the Subtractive Clustering based CPCA

Contents

2.1 Introduction

This chapter investigates the issues in the compression of geometry data of isomorphic 3D animations. The main goal is to compress the time varying geometry data across the animation frames. The connectivity data are constant throughout the animation duration satisfying the graph isomorphic property. The compression methods include the individual compression of the connectivity and the geometry data. The connectivity is compressed using a lossless algorithm, whereas the geometry data are compressed in a lossy way followed by quantization and the entropy coding. For single rate compression, the principal component analysis (PCA) is a useful tool for achieving the best compression performance as reported in the literature [7], [3], [9]. The PCA is a statistical method mainly used to reduce the dimensionality of data in the off-line mode. It takes the full animation geometry data at a time and finds the directions of the largest data variances called the principal directions or principal components (PCs). The required number of PCs, which are orthonormal to each other and form the principal axes of a new coordinate system, is selected in order of their importances. The original data are then transformed on to these principal axes to get the projected data called the PC coefficients. As the transformed data have reduced dimension compared to the original data, the PCA is basically a lossy compression technique. However, the amount of loss can be controlled by required reconstruction quality. In this chapter, we have applied the PCA on the clusters of vertex trajectories for dimensionality reduction of geometry data. The work mainly investigates on improving the performance of the clustered PCA (CPCA) algorithms. To initialize the clustering process with stable cluster centres, a density function based detection of initial cluster centre trajectories is proposed.

The rest of the chapter is organised as follows: Section 2.2 gives an overview of the PCA based animation geometry compression methods. In Section 2.3, an overview of the algorithms that have integrated clustering technique along with the PCA method for animation geometry compression is presented. The details of the CPCA algorithm and the issues associated with the improvement of the CPCA algorithm are also discussed. In Section 2.4, the details about the proposed subtractive clustering based CPCA (S-CPCA) method are presented. Section 2.5 presents experimental results and the concluding remarks are made in Section 2.6.

2.2 Animation geometry compression using the PCA

There are a number of variants of PCA based compression methods for 3D animation geometry. Alexa and Müller [7] pioneered the use of the PCA for compressing the geometry of a 3D animation. The PCA is applied in the temporal direction of the frame-based animation geometry matrix $\mathbf{A}$ (given by Equation (1.6)) to represent each frame as a linear combination of a set of uncorrelated basis vectors called eigen shapes (eigenframes). Before the application of the PCA, all the frames are normalized by translating their centres of masses to the origin. The compression scheme is lossy and the amount of loss is controlled by the required compression ratio or the visual reconstruction quality.

Consider the animation geometry matrix $\mathbf{A} = \left[\mathbf{a}_1 \,|\cdots|\, \mathbf{a}_i \,|\cdots|\, \mathbf{a}_{N_f}\right]_{3N_v \times N_f}$ with N_v number of vertices and N_f number of frames as given by Equation (1.6). The algorithmic steps for the PCA based compression of $\mathbf{A}$ matrix are given below.

A. **PCA encoding steps**

(i) *Compute the mean frame vector* $\mathbf{m}_A$ *of the geometry matrix* $\mathbf{A}$ *as*

$$\mathbf{m}_A = \frac{1}{N_f} \sum_{i=1}^{N_f} \mathbf{a}_i \tag{2.1}$$

(ii) *Subtract the mean frame vector* $\mathbf{m}_A$ *from each frame of* $\mathbf{A}$ *to get the resulting mean centred matrix* $\mathbf{A}_m$ *as given by*

$$\mathbf{A}_m = \left[(\mathbf{a}_1 - \mathbf{m}_A) \;|\; (\mathbf{a}_2 - \mathbf{m}_A) \,|\ldots|\, (\mathbf{a}_{N_f} - \mathbf{m}_A)\right] \tag{2.2}$$

(iii) *Find the covariance matrix*

$$\mathbf{Cov}_A = \mathbf{A}_m \mathbf{A}_m^T \tag{2.3}$$

(iv) *Determine the eigenvalues and eigenvectors of* $\mathbf{Cov}_A$ *using the eigen decomposition given by*

$$\mathbf{Cov}_A = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T \tag{2.4}$$

where $\mathbf{U}$ *is a* $3N_v \times 3N_v$ *matrix whose columns represent the eigen vectors* $\mathbf{u}_i$ *of* $\mathbf{Cov}_A$ *and* $\mathbf{\Lambda}$ *is the diagonal matrix whose diagonal elements are the corresponding eigen values* λ_i*.*

(v) *Arrange the eigenvectors* $\mathbf{u}_i$ *in descending order of their corresponding eigen values* λ_i *such*

that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{3N_v}$. *Select the first* c ($c << 3N_v$) *eigen vectors of* $\mathbf{Cov}_A$ *as per the compression requirement. These* c *eigen vectors are the PCs.*

(vi) *Find the PCA transformed coefficient matrix* $\mathbf{Y}$ *of size* $(c \times N_f)$ *as given by*

$$\mathbf{Y} = \mathbf{U}_c^T \mathbf{A}_m \tag{2.5}$$

where $\mathbf{U_c}$ *is a* $(3N_v \times c)$ *matrix consisting of the first* c *PCs of* $\mathbf{Cov}_A$.

(vii) *Quantize the elements of mean vector* $\mathbf{m}_A$, *eigenvectors* $\mathbf{U}_c$ *and the PCA coefficient matrix* $\mathbf{Y}$.

B. **PCA decoding steps**

(i) *De-quantize the mean vector* $\mathbf{m}_A$, *the eigenvectors matrix* $\mathbf{U}_c$ *and the coefficient matrix* $\mathbf{Y}$ *from the encoded bit-stream and reconstruct those as* $\tilde{\mathbf{m}}_A$, $\tilde{\mathbf{U}}_c$ *and* $\tilde{\mathbf{Y}}$ *respectively.*

(ii) *Form the mean matrix* $\tilde{\boldsymbol{\mu}}_A$ *using the reconstructed mean vector* $\tilde{\mathbf{m}}_A$ *as*

$$\tilde{\boldsymbol{\mu}}_A = [\tilde{\mathbf{m}}_A \,|\tilde{\mathbf{m}}_A| \cdots |\, \tilde{\mathbf{m}}_A]_{3N_v \times N_f} \tag{2.6}$$

(iii) *Obtain the reconstructed animation geometry matrix,* $\tilde{\mathbf{A}}$, *as*

$$\tilde{\mathbf{A}} = \tilde{\mathbf{U}}_\mathbf{c}\tilde{\mathbf{Y}} + \tilde{\boldsymbol{\mu}}_A \tag{2.7}$$

2.2.1 Enhancing the PCA performance

The above PCA based compression scheme does not exploit the temporal coherence among the vertex trajectories across the frames and hence has poor compression performance. Karni and Gotsman [3] enhanced the method of [7] by combining the PCA along with linear predictive coding (LPC). The PCA is applied on the animation geometry matrix $\mathbf{A}$ across the frames to exploit the spatial correlations. The results are a set of PCs (eigenframes) and their corresponding PC coefficients. The second order LPC was applied on the PC coefficients in the temporal direction to exploit the large temporal coherence present in the sequence. The PCA-LPC algorithm gives better geometry compression results for the animation sequences that have larger number of frames than the number of vertices per frame. The disadvantages of the methods are the large memory required for storing the

autocorrelation matrix given by Equation (2.4) and the significant processing time required for computing the PCs and PC coefficients from $\mathbf{Cov}_A$. The size of $\mathbf{Cov}_A$ is $(3N_v \times 3N_v)$. So, the above frame-based PCA compression methods are not suitable for the animation sequences having hundreds of frames and thousands of vertices.

Some of the existing compression schemes have applied the PCA on trajectory space of the animation geometry matrix $\mathbf{B}$ (given by Equation (1.8) in Chapter 1) to exploit the temporal coherence. Vasa and Skala proposed the Coddyac [34] algorithm by applying the PCA on the trajectory space of the animation geometry followed by the prediction of the PC coefficients using the parallelogram local predictor [24]. The prediction residuals are subsequently quantized and entropy coded. The PCA basis vectors are encoded without quantization. The algorithm gives better RD performance compared to the PCA-LPC method [3]. The Coddyac algorithm is suitable only for small duration animation sequences having large number of vertices per frame. An efficient PCA based compression algorithm called COBRA (Compression of the Basis for PCA Represented Animations) [9] was also proposed by the same authors to improve the compression performance of [34]. The method uses prediction coding for the PCA basis vectors along with a non-uniform quantization scheme. As in [34], the PC coefficients are also predicted using the parallelogram predictor. The prediction residuals are quantized and encoded using the Huffman entropy coder [11]. In [51], the authors suggested further improvement for the PCA based compression schemes by proposing two geometric predictors namely, the least square predictor (LSP) and the radial basis function predictor (RBFP). The predictors could exploit the knowledge about of the geometrical information of the data to provide more accurate prediction and compact representation. An optimized mesh traversal algorithm for the PCA based dynamic mesh compression methods was proposed by Vasa [52] to achieve significant data reduction.

There are a few compression methods proposed by combining the clustering technique with the PCA algorithm. Sattler et al. [8] proposed to combine the clustering technique with the PCA, which is called the clustered PCA (CPCA). The initial step is to make spatial clusters of the vertex trajectories stored in the animation geometry matrix $\mathbf{B}$ using the Lloyd's algorithm [53]. The PCA is then applied individually on each cluster to select the required number of eigen vectors (eigen trajectories) and the corresponding PC coefficients per cluster. The CPCA scheme is reported to give better results than the PCA-based and PCA-LPC methods in terms of the animation reconstruction error. Amjoun and

Straßer [43] developed a new local PCA (LPCA)-based animation compression method by clustering the trajectories using a local similarity measure and expressing each trajectory in a local coordinate frame defined for each cluster. Additionally, a bit allocation scheme is applied to assign more bits to clusters where more basis vectors and PC coefficients are required to achieve the desired precision. Results indicate that the LPCA has better compression performance than the CPCA and other previous methods for animation sequences with large number of vertices per frame. In [54], Luo et al. presented a PCA based compression method for 3D mesh sequences using temporal segmentation of the frames. The frames are grouped into clusters of varying lengths based on their pose similarity after eliminating the global rigid transformation from each frame. The PCA is applied on each cluster to represent the frame coordinates in a reduced dimension. An intra-cluster compression method using linear coding based on pose similarity (LCPS) is used to achieve minimum reconstruction error by exploiting the temporal coherence among the PCA transformed motion fragments within the clusters. This compression method is beneficial for 3D mesh sequences which have larger number of frames than the number of vertices and contain frame data with repetitive motion over the sequence.

The literature review suggests that many of the existing methods for animation geometry compression use the PCA as the important block for compression. The PCA approaches are efficient because it analyses the coherence of the entire sequence. They are best known as global compression methods and give best compression results when processed off-line on the 3D animation data. As per the survey, it is found that compared to the direct PCA, the PCA on clusters provides better compression of animation geometry by exploiting both the spatial and temporal coherence across the sequences. Because clustering makes grouping of the vertices of frames such that those belonging to the same cluster have similar motion trajectories or shapes. Moreover, in the clustering based PCA methods, the number of PCs required to describe the motions or shapes in each cluster is usually much smaller than the number of PCs without clustering. This work also aims at investigating the potential of the PCA as an animation compression tool when combined with better clustering methods. The details of the CPCA algorithm are presented in the subsequent sections.

2.3 Animation geometry compression using the CPCA

In the CPCA method [8], the vertex trajectories in the animation geometry matrix **B** are grouped into clusters based on their minimum reconstruction errors with respect to the PCs obtained applying the PCA of each cluster. The method uses a combination of clustering and the PCA on the clusters in each iteration and selects the final clusters for the vertex trajectories using a reconstruction error metric. The steps of the CPCA method is summarized below:

1) For the given parameters K and c, initialize a set of K cluster-centre trajectories $\mathbf{X}_c = \left\{\mathbf{t}_{c_j}, j = 1, \ldots, K\right\}$ randomly from the set of N_v vertex trajectories in the $3N_f$- dimensional space. Assign a set of c orthonormal vectors, $\mathbf{U}^j = \left\{\mathbf{u}_h^j \in \mathbb{R}^{3N_f}, h = 1, \cdots, c,\ j = 1, \cdots, K\right\}$ to each cluster j.

2) Group the vertex trajectories $\mathbf{t}_i \in \mathbb{R}^{3N_f}, i = 1, \cdots, N_v$ into K sets of cluster, $\mathbf{\Phi}^j$, by assigning them into the closest cluster-centre $\mathbf{t}_{c_j}$. The distance to a cluster-centre $\mathbf{t}_{c_j}$ is based on the squared reconstruction error and given by

$$\left\|\mathbf{t}_i - \tilde{\mathbf{t}}_i^j\right\|^2 = \left\|\mathbf{t}_i - \mathbf{t}_{\mathbf{c}_j} - \sum_{h=1}^{c} \left\langle \mathbf{t}_i - \mathbf{t}_{\mathbf{c}_j}, \mathbf{u}_h^j \right\rangle \mathbf{u}_h^j \right\|^2, i = 1, \cdots, N_v, j = 1, \cdots, k. \tag{2.8}$$

where $\tilde{\mathbf{t}}_i^j$ is the reconstructed vertex trajectory with respect to the basis vectors of cluster j.

3) Compute the new cluster centre trajectories $\mathbf{t}_{\mathbf{c}_j}$ by averaging the vertex trajectories in $\mathbf{\Phi}^j$.

4) Apply the PCA on each cluster and select a new set of c basis vectors (*eigen trajectories*) $\mathbf{U}^j$ per cluster based on their eigen values.

5) Iterate steps 2 to 4 until the change in average reconstruction error of the vertex trajectories falls below a given threshold.

6) For each trajectory $\mathbf{t}_i$, store the cluster index j in an array of cluster index $I_c(i)$ which it belongs to.

7) For each cluster j,

a) Project the vertex trajectories $\mathbf{\Phi}^j$ onto the subspace spanned by basis vectors $\mathbf{U}^j$ and get the PC coefficients $\mathbf{Y}^j$.

b) Quantize the elements of $\mathbf{U}^j$, $\mathbf{Y}^j$ and mean trajectory vector $\mathbf{t}_{\mathbf{c}_j}$ by using a uniform quantizer.

8) Encode the the cluster index vector $\mathbf{I}_c$ and the quantization indices from the clusters using an entropy coder to get the compressed geometry data.

9) Encode the connectivity information separately using the Edgebreaker [30] connectivity compression algorithm.

The success of the above CPCA algorithm mainly depends on the effectiveness of the clustering process. We have observed the following issues associated with the CPCA algorithm as discussed below:

(i) In the CPCA algorithm, the clustering process starts with the random selection of initial cluster centres. Because of this, the CPCA algorithm does not necessarily give the optimal partitions of the vertex trajectories in each run, as it may converges to numerous local minima instead of the global minimum. So, it gives different compression performance at different runs of the algorithm on the same animation data. Therefore, there is a scope for improving clustering process by proper initialization of cluster centres instead of random initialization. This may give better results in terms of compression performance.

(ii) The number of clusters are manually decided in the case of the CPCA. There is a scope for automatically select the number of clusters for the optimal performance using a cluster validity index parameter.

(iii) The CPCA algorithm uses a uniform quantization scheme for quantizing the elements of PCs and PC coefficients for each cluster. The performance can be further improved by implementing a non-uniform bit-allocation scheme for the PCs and the PC coefficients belonging to each cluster.

Based on the above observations, this chapter aims at carrying out the following investigations for improving the performance of the CPCA algorithm with better clustering of vertex trajectories.

(i) The random initialization of the cluster centres in the CPCA algorithm is proposed to be replaced by a stable initialization method of cluster centres using the subtractive clustering (SC) technique [55]. This will lead to stable compression performance in each run of the algorithm.

(ii) A non-uniform quantization scheme has been applied to select a set of quantization bits for each cluster based on the RD criteria. The set of quantization bits are used to quantize the elements of PCs, PC coefficients and the mean trajectory vector in each cluster.

(iii) An improved arithmetic coding is proposed for the lossless encoding of the non-uniform quantization levels.

2.4 Proposed subtractive clustering based CPCA method

This section presents a novel subtractive clustering (SC) based CPCA (S-CPCA) method using K-means clustering combined with the SC algorithm. The SC algorithm is applied to initialize the K-means clustering process with stable cluster centres. A non-uniform quantizer is used to quantize the elements of PCs and PC coefficients belonging to each cluster. The quantized levels of PCs and PC coefficients in each cluster are encoded with a proposed block based arithmetic coding to get better compression gain. The details about the SC algorithm, the non-uniform quantizer and block-based arithmetic coder are presented.

Subtractive clustering of vertex trajectories

The SC algorithm is a density function based unsupervised data analysis tool. It finds the cluster centres based on the density values (also called potential values) of the data points in the feature space. It considers each data point as the candidate for a potential cluster centre. The potential value at each data point is calculated based on its distance from the other data points. The data point with the highest potential value is selected as the first cluster centre. After that, the potentials of all the data points within a predefined radius of the detected cluster centre are removed. The algorithm then finds the point with the next highest potential value and selects that as the next cluster centre. This procedure is repeated until a predefined criterion is met or the desired number of clusters are obtained. The SC algorithm is a fast, one-pass algorithm for estimating the number of clusters and provides a set of cluster centres from the data set for some input parameters. The estimated cluster centres can be either directly used for approximate clustering purpose or they can be further used as initial cluster centres for the iterative optimization based clustering methods like K-means [56], fuzzy K-means(FKM) [57], [58] and particle swarm optimization (PSO) [59] etc. The mathematical details

of the SC algorithm is presented below.

Consider the set of N_v vertex trajectories $\{\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_{N_v}\} \in \mathbb{R}^{3N_f}$ in the $3N_f$-dimensional space of the animation matrix $\mathbf{B}$ as the set of data points. To find a set of K cluster centre trajectories, $\{\mathbf{t_{c_1}}, \mathbf{t_{c_2}}, \ldots, \mathbf{t_{c_K}}\} \in \mathbb{R}^{3N_f}$, the trajectory vectors are normalized in each dimension using $t_i^d = \frac{t_i^d - \min\{\mathbf{t}^d\}}{\min\{\mathbf{t}^d\} - \max\{\mathbf{t}^d\}}, i = 1, \cdots, N_v, \quad d = 1, \cdots, 3N_f$ such that their coordinate ranges are equal in each dimension, i.e. they are bounded by a hypercube. Since each trajectory vector is a candidate for cluster centres, the potential value at a vertex trajectory $\mathbf{t_i}$ is computed as

$$P_i = \sum_{j=1}^{N_v} \exp\left(\frac{-\|\mathbf{t}_i - \mathbf{t}_j\|^2}{\left(\frac{r_a}{2}\right)^2}\right), i = 1, \cdots, N_v. \tag{2.9}$$

where $\|\mathbf{t}_i - \mathbf{t}_j\|^2$ is the square of the distance between any two vectors $\mathbf{t_i}$ and $\mathbf{t}_j$ and $r_a > 0$ is a constant. Since the potential value calculated for each trajectory is a function of its distances from all other trajectories, a vertex with many neighbourhood vertices which are moving coherently across whole animation frames will have high potential value. The constant r_a is effectively a radius of influence of the neighbourhood vertex trajectories on the potential value. After calculating the potential at all the trajectories, a cluster centre $\mathbf{t_{c}}_k$ is selected at the trajectory vector which has the highest potential value and its potential value $P_{c_k}^*$ is given by

$$P_{c_k}^* = \max_i \{P_i\}, i = 1, \cdots, N_v \tag{2.10}$$

The potentials of all the trajectories are subsequently modified as

$$P_i = P_i - P_{c_k}^* \exp\left(\frac{-\|\mathbf{t}_i - \mathbf{t_{c_k}}\|^2}{\left(\frac{r_b}{2}\right)^2}\right), i = 1, \cdots, N_v \tag{2.11}$$

where $r_b > 0$ is a constant defining a neighbouring radius such that trajectories within this radius will have measurable reduction in the potential value. Because of this modification, the vertex trajectories near the cluster centre $\mathbf{t_{c_k}}$ will have significantly reduced potential and are very unlikely to be selected as the cluster centre during next iteration. The value of r_b is normally larger than r_a to avoid closely placed cluster centres and it is generally taken greater than r_a or $1.5r_a$ as suggested in [55]. After revising the potential of all the vertex trajectories, the next cluster centre $\mathbf{t_{c_{k+1}}}$ is selected at the vertex trajectory having the highest among remaining potential values. The effects of this cluster centre $\mathbf{t_{c_{k+1}}}$ are again removed using Equation (2.11). This subtractive iterative process is repeated

until sufficient number of vertex trajectories are obtained as the cluster centres for an input threshold δ for the potential.Thus the iteration stops when

$$\frac{P^*_{c_k}}{P^*_{c_1}} < \delta. \tag{2.12}$$

where $P^*_{c_1}$ is the potential of the first cluster centre. There are other criteria for stopping clustering process as well as selecting and rejecting vertex trajectories as cluster centres based on acceptance and rejection parameters as stated in [55].

Non-uniform bit allocations

The signal energy of the original animation geometry data gets concentrated in the first few PCs after the PCA decomposition. Hence, a good approximation of the original data can be obtained by using only the first few PCs and the corresponding PC coefficients. These PCs are usually arranged in the descending order of their eigen values and the required number of PCs is selected based on an input threshold value for cumulative energy content in the eigen values. Since the elements of PCs and the PC transformed coefficients are floating-point numbers, it is necessary to quantize them prior to storage/communication. The quantization can be done either in a uniform or in a non-uniform way. In uniform quantization, the elements of all the PCs and the corresponding PC coefficients are quantized with a fixed number of bits. This process may not give the optimal RD performance since it does not consider the importance of the PCs according to their eigen values. It will be, therefore, advantageous if the quantization bits for the elements of individual PCs are determined based on their eigen values. In this scheme, each cluster is allotted a number of bits according to the number of selected PCs in the cluster. The available bits for a cluster is non-uniformly allotted to each PC according to the corresponding eigen value. The elements of a PC and the corresponding transformed coefficients are uniformly quantized into a number of levels determined by the number of bits allocated for the PC. Heu et al. [60] developed an RD optimised bit allocation scheme for the singular vectors in an SVD-based compression scheme for the animation geometry. According to this scheme, the quantization step size s_i for the ith singular vector is given by

$$s_i = \frac{\sigma_1}{\sigma_i} s_1 \tag{2.13}$$

where s_1 is a reference step size for the first singular vector and σ_is represent the singular values. Note that the eigen value λ_i and the corresponding singular value σ_i are related by the relation

$$\sigma_i = \sqrt{\lambda_i} \tag{2.14}$$

Using Equation (2.13) and (2.14), the number of bits q_i allotted to quantize the ith PC is obtained as

$$q_i = q_1 - \left\lceil \frac{1}{2} \log_2 \left(\frac{\lambda_1}{\lambda_i} \right) \right\rceil \tag{2.15}$$

Each element of ith PC and the corresponding transformed vector are uniformly quantized with q_i bits.

Proposed block arithmetic coding for quantization levels

To get better coding efficiency in terms of transmission and storage, the quantized levels of the PCs and the corresponding PC coefficients are encoded in lossless manner using an entropy encoder. The application of an entropy coding helps in achieving additional compression in terms of bitrates by encoding the quantization levels more compactly based on their statistical characteristics. In general, Huffman and arithmetic codings are used in the entropy coding stage. The advantage of an arithmetic encoder compared to a Huffman encoder is that it operates on a group of symbols and assigns fractional bits to the symbols based on their probabilities. On the other hand, a discrete number of bits are used for each symbol in the Huffman encoder. In this work, we have proposed a block based arithmetic coding for the quantization levels of the PCs and PC coefficients of the vertex trajectories in each cluster. The scheme works as follows:

(i) For each cluster, group together the quantization levels of the PCs which are quantized with same q_i as a block. The number of blocks for each cluster is equal to the number of different non-uniform quantization bits assigned for that cluster. Encode the levels in each block in a lossless manner by applying the arithmetic coding.

(ii) Similarly, group together the quantization levels of the PC coefficients in each cluster which are quantized with same q_i as a block and encode the levels in each block by applying the arithmetic coding.

The block based arithmetic encoding exploits the redundancy in quantization levels in a better way

compared to the arithmetic encoding of all the quantization levels together per cluster.

2.4.1 S-CPCA encoder

The block diagram of the encoder and the decoder for the proposed S-CPCA method is shown in Figure 2.1.

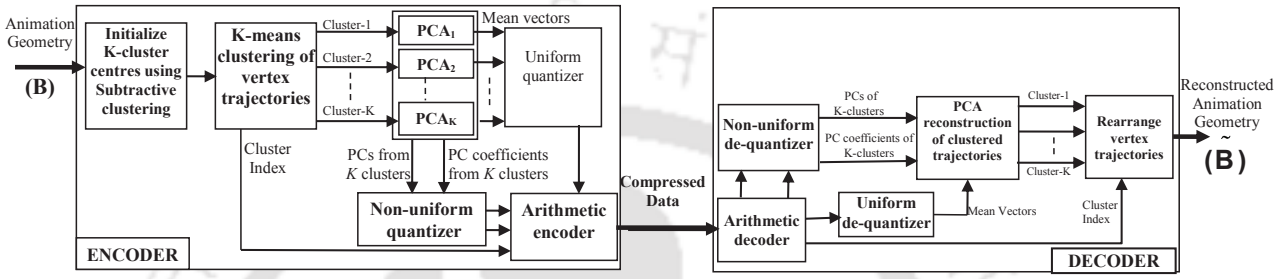


Figure 2.1: Block diagram of the encoder and the decoder of proposed S-CPCA method

The trajectory based animation geometry matrix **B** is given as the input to the encoder. The encoder uses the SC algorithm to select K vertex trajectories as the initial cluster centres. The stable cluster centres obtained from the SC algorithm are used to initialize the K-means clustering algorithm to group the vertex trajectories based on the temporal information of the vertex positions across the frames. The K-means algorithm optimizes the clustering of vertex trajectories based on an objective cost function. The PCA is then applied individually on each cluster to select the optimum number of eigen trajectories or PCs. The selection of the optimum number of PCs per cluster is decided by the input threshold value for the cumulative energy content in the eigen values. The vertex trajectories belonging to each cluster are then projected onto the space spanned by the PCs to get the PC coefficients. The set of PCs, PC coefficients and the mean trajectory vector for each cluster are quantized using the proposed non-uniform quantizer. The algorithm steps for the proposed S-CPCA method are described below:

Algorithmic steps for the S-CPCA method

(i) Given the number of clusters K, compute the density or potential value P_i at each vertex trajectory $\mathbf{t}_i, i = 1, \ldots, N_v$ using the Eq. (2.9).

(ii) Initialize k=1

(iii) Select the cluster centre trajectory $\mathbf{t}_{c_k}$ as the trajectory having the largest potential value $P^*_{c_k}$ as given by the Eq. (2.10).

(iv) Revise the potential value P_i of each vertex trajectory $\mathbf{t}_i$ using Equation (2.11).

(v) Increment k by 1 and continue this subtractive process to find the next cluster centres by iterating steps (iii) and (iv) until K cluster centres are obtained.

(vi) Apply the K-means clustering algorithm on the vertex trajectories with the above cluster centres as the initial cluster centres to get the optimum clusters of vertex trajectories and the cluster index vector. The similarity between two vertex trajectories is measured by their Euclidean distances from the cluster centres instead of the reconstruction error metric for the trajectories as used in the CPCA algorithm [8]. The steps for the K-mean based clustered PCA are as follow:

(a) Initialize the K-cluster centre trajectories $\mathbf{t}_{\mathbf{c}_j}$, $j = 1, \ldots, K$ as the output of step (v).

(b) Find the square of the Euclidean distances of all the N_v trajectories from each of the centre trajectories $\mathbf{t}_{\mathbf{c}_j}$ and put them in a distance matrix $\mathbf{D_t}$ given by

$$\mathbf{D_t} = [d_{ij}] = \left[\left\| \mathbf{t}_i - \mathbf{t}_{\mathbf{c}_j} \right\|^2 \right] , \; i = 1, \ldots, N_v \tag{2.16}$$

(c) Assign each trajectory $\mathbf{t}_i$ to its nearest cluster centre based on minimum distance from the matrix $\mathbf{D_t}$ and store the cluster number it belongs to in an array of cluster index $\mathbf{I}_c$ given by

$$I_c(i) = \arg_j \{\min [d_{ij}]\} \tag{2.17}$$

(d) Update the vertex trajectories in each cluster $\mathbf{\Phi}^j, j = 1, \ldots, K$ using the cluster index $\mathbf{I}_c$ as given by

$$\mathbf{\Phi}^j = \{\mathbf{t}_{I_c(i)} \mid I_c(i) = j\}, \;\; i \in \{1, \ldots, N_v\} \tag{2.18}$$

(e) Update the cluster centres $\mathbf{t}_{c_j}$ with the mean of the vertex trajectories belonging to that cluster as given by

$$\mathbf{t}_{c_j} = \frac{1}{|\mathbf{\Phi}^j|} \sum_{m \in \mathbf{\Phi}^j} \mathbf{t}_m \tag{2.19}$$

where $\left|\mathbf{\Phi}^j\right|$ is the number of vertex trajectories in cluster $\mathbf{\Phi}^j$.

(f) Repeat the above steps (b), (c), (d) and (e) until there is no change in the cluster index $\mathbf{I}_c$.

(vii) Apply the PCA on each cluster $\mathbf{\Phi}^j$ and select the required number of PCs per cluster, $\mathbf{U}^j$, based on the given threshold energy value, λ_{Th}, representing the percentage of cumulative energy contained in the ordered eigen values.

(viii) Compute the PC transformed coefficients per cluster, $\mathbf{Y}^j$, by projecting the vertex trajectories on the space spanned by the PCs in $\mathbf{U}^j$.

(ix) Allocate a set of non-uniform quantization bits for the PCs in $\mathbf{U}^j$ and the corresponding PC coefficients in $\mathbf{Y}^j$ for each cluster j using Equation (2.15). Quantize the elements of each PC and the corresponding PC coefficients uniformly with the allotted quantization bits. Encode the sets of non-uniform quantization bits for each cluster and send it to the decoder.

(x) Group the quantization levels of the PCs quantized with same quantization bit for each cluster and encode them using block arithmetic coding to gain compression efficiency. Similarly, quantization levels for PC coefficients per cluster are grouped and encoded with block arithmetic coding.

(xi) Quantize the elements of mean trajectory vectors $\mathbf{t}_{c_j}$ for each cluster using an uniform quantizer and encode the quantization levels using the arithmetic encoder.

(xii) Encode the vector $\mathbf{I}_c$ representing the cluster indices of the vertex trajectories using an arithmetic encoder and send along the compressed bit-stream.

2.4.2 S-CPCA decoder

The decoder reads the compressed bitstream from the encoder and performs the following steps to reconstruct the animation geometry data.

(i) Decode the group of quantization levels of the elements of PCs and the PC coefficients for each cluster by applying the arithmetic decoder in blocks.

(ii) Perform de-quantization of the elements of the PCs and PC coefficients for all the clusters using the non-uniform quantization bits assigned for them.

(iii) Decode the quantization levels of the elements of mean trajectory vectors for all the clusters using the arithmetic decoder and de-quantize them uniformly.

(iv) Reconstruct the vertex trajectories for each cluster by applying the PCA decoding steps on the de-quantize PCs, PC coefficients and the mean trajectory vectors.

(v) Decode the cluster index vector using the arithmetic decoder and re-arrange all the vertex trajectories based on the cluster index vector to get the reconstructed animation geometry matrix $\widetilde{\mathbf{B}}$.

2.5 Experimental results

2.5.1 Details of animation sequence

To study the performances of the proposed S-CPCA method, standard test 3D animation sequences are considered. The details about the sequences are shown in Table 2.1.

Table 2.1: The details of 3D animation sequences

Name	**Vertices** (N_v)	**Triangles** (N_t)	**Frames** (N_f)
Face	757	1468	950
Cow	2904	5804	204
Dolphin	6179	12337	101
Chicken	3030	5664	400
Dance	7061	14118	201
Snake	9179	18354	134
Jump	15830	31660	222
Cloth	9987	19494	200
MocapDance	14409	28648	263

All these sequences are isomorphic animation sequences with constant connectivity across the frames. The frame-wise geometry and connectivity data for the 3D animation are read and stored in matrix $\mathbf{B}$ and $\mathbf{C_d}$ respectively. The connectivity information is compressed in lossless manner using

the TFAN encoder discussed in Section 1.3.6. The proposed algorithm mainly focuses on compressing the geometry data in **B**. Experiments are carried out to evaluate the compression performance of the proposed method using the widely used distortion metrics such as KG_{err} and $STED$ as presented in Section 1.4. The values of these metrics are evaluated against different target $bpvf$ values to get RD performances for the proposed method.

Experimental results for random-initialization CPCA method

In the first set of experiments, we apply the existing CPCA algorithm on different 3D animation sequences with the random initialization of the cluster centres. A uniform quantization factor of $q = 16$ bits to quantize the elements of PCs, PC coefficients and the mean trajectory vector of each cluster. The performance is measured in terms of the CR, the KG_{err} and the $bpvf$. Table 2.2 shows the result of applying this method on the "Chicken" sequence.

Table 2.2: Results using CPCA method (random initialization) on "Chicken" sequence

(a) Results using energy threshold per cluster

Selection of eigen vectors per cluster based on total energy threshold of $\lambda_{Th} = 99.9\%$					
Run No.	No. of clusters (K)	Total Eigen vectors	CR	KG_{err}	$bpvf$
1	5	64	28.838	0.187	1.652
2	5	61	30.399	0.188	1.566
3	5	59	30.238	0.159	1.575
4	5	64	28.838	0.187	1.652
5	5	58	30.969	0.173	1.537

(b) Results using fixed no. of eigen vectors

Selection of fixed no. of eigen vectors per cluster (c=12)					
Run No.	No. of clusters (K)	Total Eigen vectors	CR	KG_{err}	$bpvf$
1	5	50	36.605	0.425	1.289
2	5	50	36.605	0.392	1.289
3	5	50	36.605	0.484	1.289
4	5	50	36.605	0.388	1.289
5	5	50	36.605	0.495	1.289

In Table 2.2(a), the results are shown by selecting the eigen vectors per cluster based on cumulative energy threshold value of $\lambda_{Th} = 99.9\%$. The input cluster value is $K = 5$ for which the algorithm is executed for 5 runs. At the first run, the total number of eigen vectors are 64 for the above energy threshold value. This run results in CR value of 28.838, KG_{err} value of 0.187 and $bpvf$ value of 1.652. The minimum KG_{err} value 0.159 is obtained at the third run which has selected total 59 eigen vectors across the clusters resulting CR value of 30.238 with $bpvf$ value of 1.575. In Table 2.2(b), fixed number of eigen vectors are chosen per cluster and the metrics CR, KG_{err} and $bpvf$ are evaluated for 5 runs. The lowest KG_{err} value 0.388 is obtained at the fourth run which gives CR value of 36.605 with the corresponding $bpvf$ value of 1.289 for total number of eigen vectors 50. These results for 5 runs demonstrate the unstable compression results of the CPCA method. This is due to random

initialization of the cluster centres in each run.

2.5.2 Experimental results for S-CPCA method

In this section, the proposed S-CPCA method is evaluated on different 3D animation sequences to evaluate the compression performance.

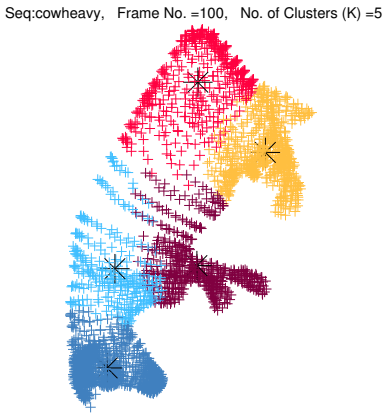
Experimental results for S-CPCA method using uniform quantizer

Table 2.3 shows the results obtained using the proposed S-CPCA method without the proposed bit-allocation and the block arithmetic coding steps on the "Cow" animation.

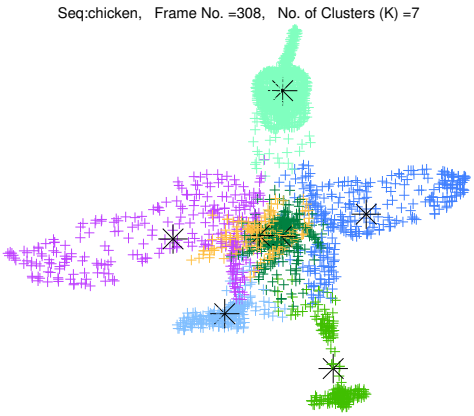
Table 2.3: Stable performance results using the S-CPCA method

Selection of eigen vectors per cluster based on $\lambda_{Th} = 99.9\%$					
Run	No. of clusters (K)	EVs per cluster	CR	$bpvf$	KG_{err}
1	5	14, 12, 13, 14, 8	23.1506	2.2224	0.5500
2	5	14, 12, 13, 14, 8	23.1506	2.2224	0.5500
3	5	14, 12, 13, 14, 8	23.1506	2.2224	0.5500
4	5	14, 12, 13, 14, 8	23.1506	2.2224	0.5500
5	5	14, 12, 13, 14, 8	23.1506	2.2224	0.5500

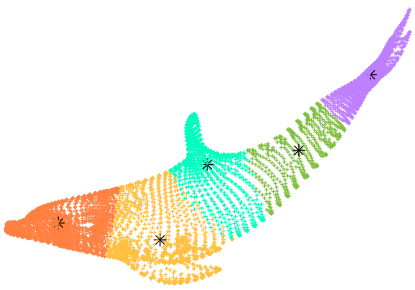
A uniform number of 16 bits are allocated to quantize the elements of PCs, the PC coefficients and the mean trajectory vector of each cluster. The number of input clusters is $K = 5$ for which the algorithm is executed for 5 runs. The threshold value of $\lambda_{Th} = 99.9\%$ selects 14, 12, 13, 14 and 8 number of PCs in the corresponding clusters. It has been observed that each run of the S-CPCA algorithm gives consistent results with KG_{err} value of 0.5500 at $bpvf$ value of 2.2224 with CR value of 23.1506. It may be due to the stable cluster centres obtained through the SC algorithm. This results in consistent clusters of vertex trajectories using the K-means algorithm. Figure 2.2 shows the frame-wise clustering results after application of the S-CPCA method using the uniform quantization method in encoding stage for "Cow", "Chicken", "Dolphin" and "Dance" animation sequences. The black stars, $*$, in each figure indicate the cluster centre for each cluster. In this case, the cluster centres will always remain same with identical cluster indices for vertex trajectories in each iteration. This is because of finding out the stable cluster centres based on the density function. This is the advantage



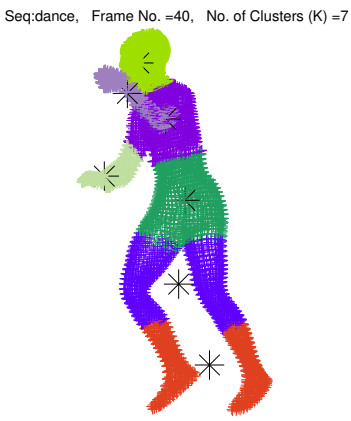
(a) 'Cow' with K=5



(b) 'Chicken' with K=7



(c) 'Dolphin' with K=5



(d) 'Dance' with K=7

Figure 2.2: Frames from different animation sequences showing the clustering results

Table 2.4: Comparative results using proposed S-CPCA method and the CPCA method

Proposed S-CPCA method					**CPCA method**		
No. of clusters (K)	c per cluster	CR	KG_{err}	$bpvf$	CR	KG_{err}	$bpvf$
"Dolphin"							
2	10	25.08	0.029	1.86	25.08	0.029	1.86
3	10	24.01	0.015	1.95	24.01	0.015	1.95
4	10	23.03	0.009	2.03	23.03	0.010	2.03
"Chicken"							
5	5	67.11	**1.062**	0.70	67.11	1.064	0.70
5	10	36.61	**0.425**	1.30	36.61	0.451	1.30
5	20	19.17	**0.038**	2.49	19.17	0.091	2.49
"Cow"							
5	10	27.09	**0.748**	1.75	27.09	0.854	1.75
5	20	14.19	**0.291**	3.36	14.19	0.376	3.36
10	10	17.90	0.484	2.65	17.90	**0.472**	2.65

of obtaining stable centres using the proposed S-CPCA method compared to the random initialization of the cluster centres in the CPCA method.

The performance metrics obtained using the proposed S-CPCA method with the uniform quantization bits for different number of clusters (K) and eigenvectors per cluster (c) are listed in Table 2.4. In this experiment, we have taken $r_a = 0.6325$, $r_b = 1.414r_a$, and a uniform quantization factor of $q = 16$ bits to quantize the PCs, PC coefficients and the mean trajectory vector of each cluster. We have also shown the results obtained using the CPCA method (average from 20 runs) for comparison. From the results, it has been observed that for fixed number of PCs (c) per cluster, the proposed S-CPCA method gives better KG_{err} for the same CR and $bpvf$ compared to the CPCA method for the "Chicken", and "Cow" sequences whereas for the "Dolphin" animation sequence, it gives almost similar results as the CPCA. Figure 2.3 shows one original and one reconstructed frame each from the "Cow", "Chicken" and "Dance" animation sequences that were compressed using the proposed S-CPCA method with uniform quantizer for K=5 clusters and c=10 per cluster. The decompressed results with CR values 27.09 in the case of "Cow" sequence, 36.61 in the case of "Chicken" sequence and 38.42 in the case of "Dance" sequence show negligible visual artifacts.

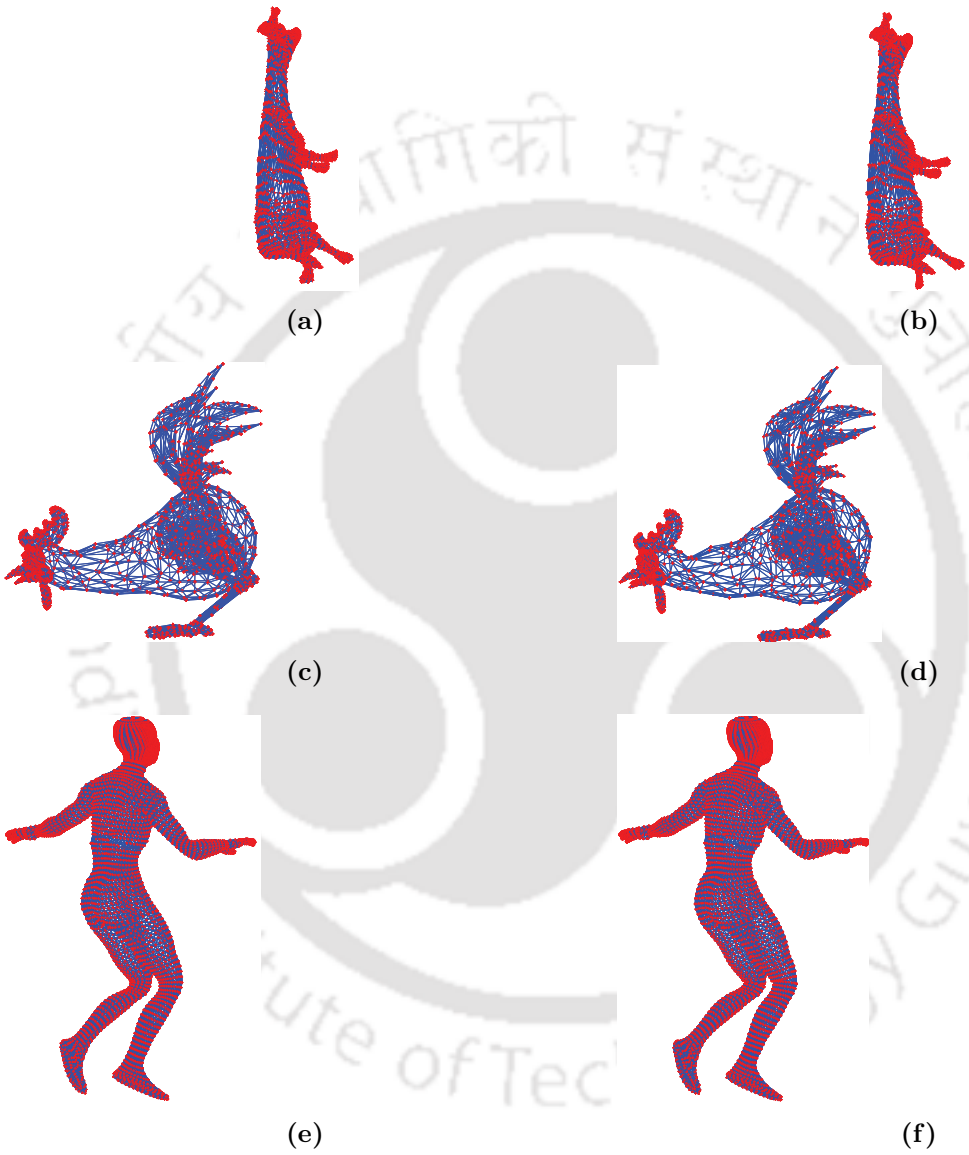
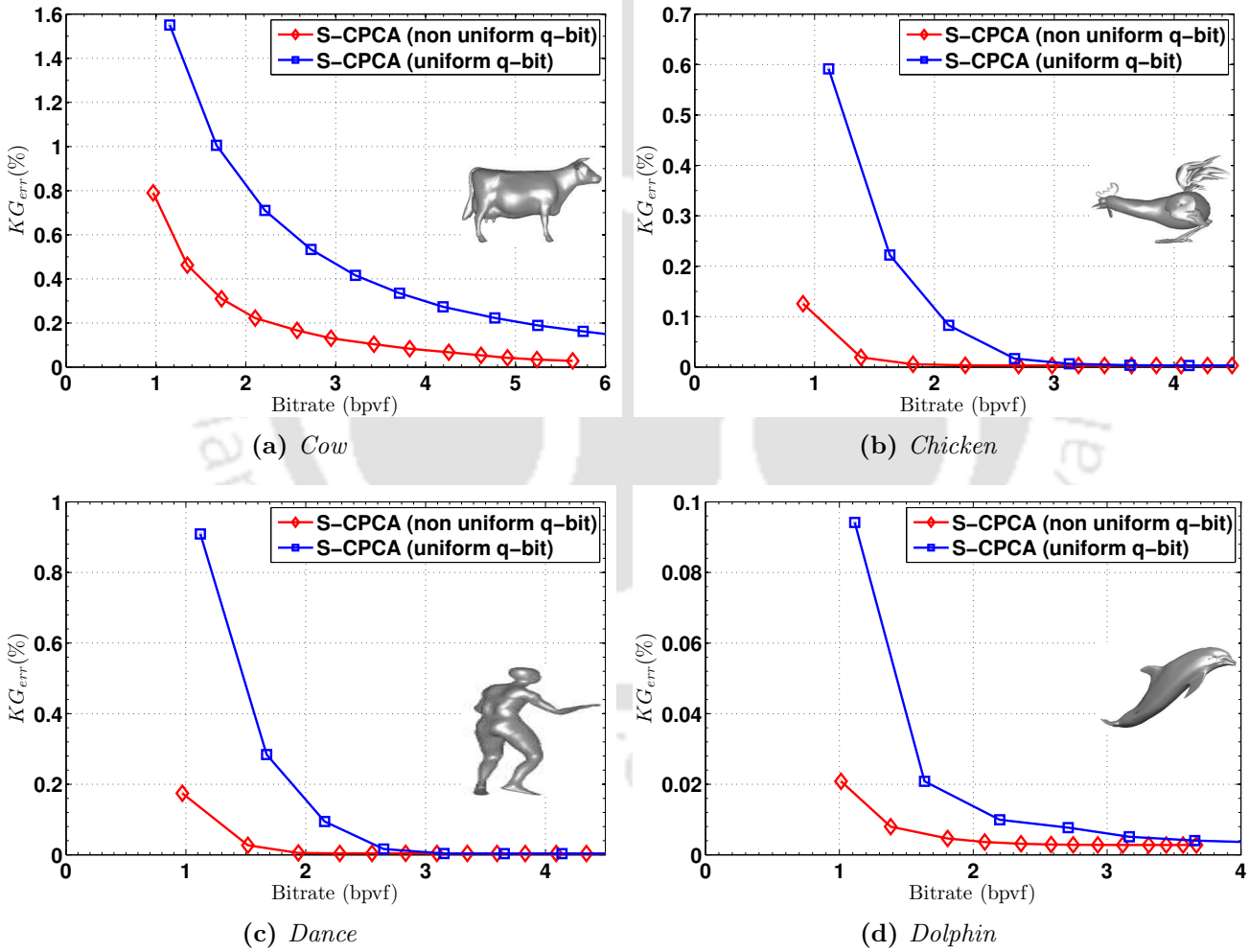


Figure 2.3: (a) Original and (b) reconstructed frames ($f = 41$) of "Cow" sequence, (c) original and (d) reconstructed frames ($f = 41$) of "Chicken" sequence, (e) original and (f) reconstructed frames ($f = 100$) of "Dance" sequence

Experimental results for S-CPCA method using non-uniform quantizer and block-based arithmetic coding

Next we apply the S-CPCA method as described in the S-CPCA algorithmic steps. The method has been evaluated using the parameters $K = 5$, $r_a = 0.6325$, $r_b = 1.414 r_a$, $\lambda_{Th} = 99.9\%$ for different target $bpvf$. The final number of PCs per cluster (c) depends on target $bpvf$. The q_{ref} for the non-uniform quantizer is varied from 16 bits to 8 bits. The experimental results are evaluated with the results of the S-CPCA method with uniform quantization. The results of both the quantization methods are shown in Figure 2.4.

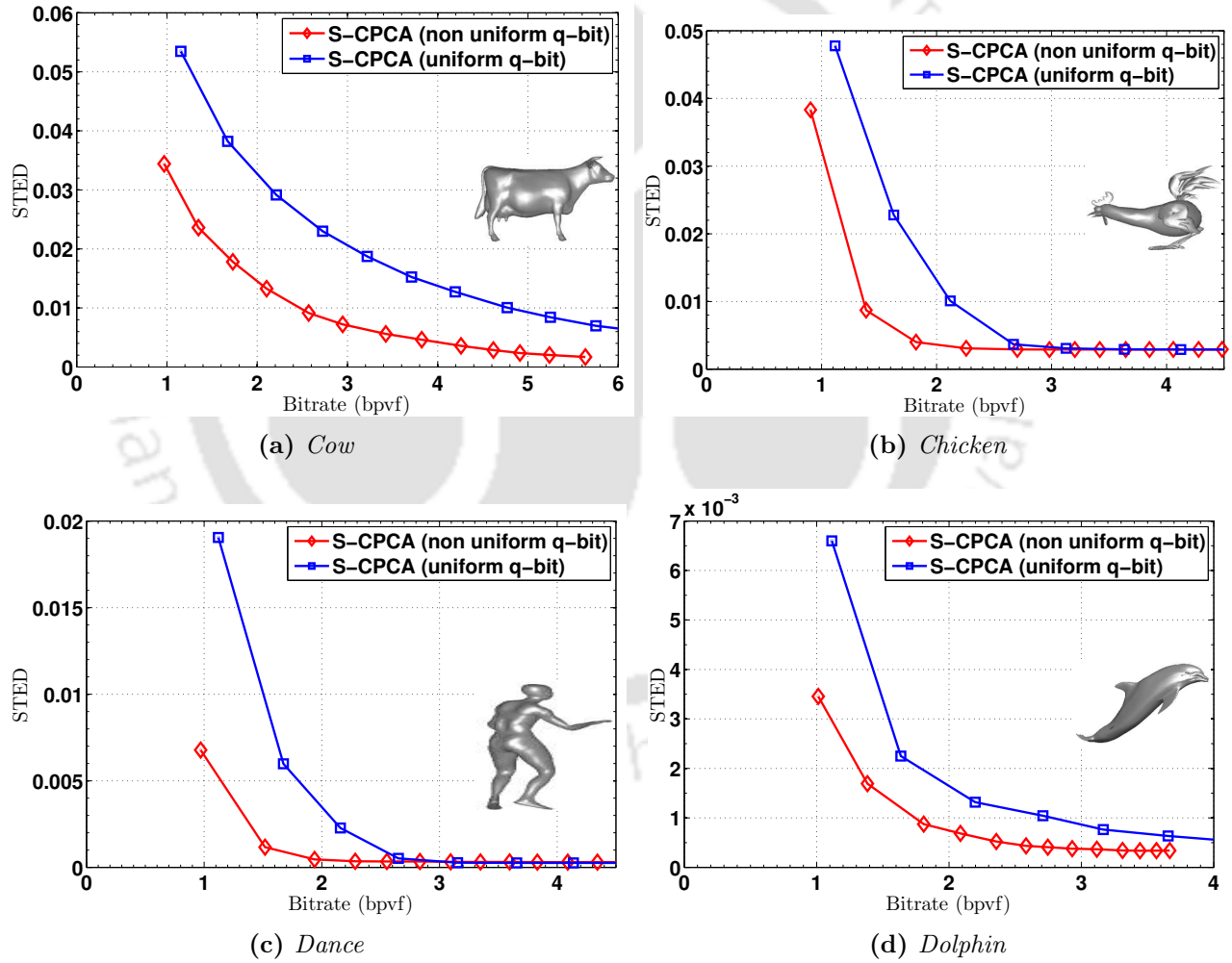


(a) *Cow* (b) *Chicken*

(c) *Dance* (d) *Dolphin*

Figure 2.4: Comparison of RD performances between uniform and non-uniform quantization method using the KG_{err} metric on a few 3D animations: (a) "Cow" sequence, (b) "Chicken" sequence, (c) "Dance" sequence and (d) "Dolphin" sequence

In Figure 2.4(a), distortion metric KG_{err} has been plotted for different target $bpvf$ values for the "Cow" sequence. The performance in terms of KG_{err} for the method with non-uniform quantization bits is found better than the uniform quantization bits for all $bpvf$ values. Similarly, for other three sequences namely, "Chicken", "Dance" and "Dolphin", the RD performances are found to be superior for the S-CPCA algorithm with non-uniform quantizer bits and the results are shown in Figures 2.4(b), 2.4(c) and 2.4(d) respectively. Thus, the proposed S-CPCA method gives better performance. To validate it further, we have also evaluated RD performances in terms of the $STED$ metric for the above four animation sequences and the results are shown in Figure 2.5.



(a) *Cow* **(b)** *Chicken*

(c) *Dance* **(d)** *Dolphin*

Figure 2.5: Comparison of RD performances between uniform and non-uniform quantization method using the $STED$ metric on a few 3D animations: (a) "Cow" sequence, (b) "Chicken" sequence, (c) "Dance" sequence and (d) "Dolphin" sequence

In Figure 2.5(a), distortion metric $STED$ has been plotted vs target $bpvf$ values for the "Cow" sequence. The performance in terms of the $STED$ metric for the S-CPCA algorithm with the non-uniform quantization bits is also found better than the S-CPCA with uniform quantization. For "Chicken", "Dance" and "Dolphin" sequences, the RD performances in terms of the $STED$ metric are also found better for the non-uniform quantization case. The respective plots are shown in Figures 2.5(b), 2.5(c) and 2.5(d). So, for both the distortion metrics (KG_{err} and $STED$), the proposed S-CPCA method results in higher compression performance.

2.5.3 Comparison of performance of S-CPCA with other methods

In this section, the performance of the proposed S-CPCA method is compared with the existing CPCA [8] and LPCA [43] methods. The results for the CPCA and the LPCA methods are taken directly from the respective papers. The comparisons are made using the KG_{err} metric obtained for an input target $bpvf$ value. Since, for a given target bpvf, the selection of exact number of clusters K depends on many factors like values of q_{ref}, N_v, N_f and c per cluster, we have applied a trial and error method to find the value of K for different animations for a target $bpvf$. It has been observed through experiments that the values of K from 5 to 8 provide good compression results for most of the animations. We have taken $K = 5$ for our methods based on the above facts. The performance comparisons are shown in Tables 2.5, 2.6 and 2.7 for the "Cow", "Chicken" and "Dolphin" sequences respectively.

Table 2.5: Comparison of proposed S-CPCA with other methods on"Cow" sequence

Method	"Cow" Sequence			
	K	**c per cluster**	**KG_{err}**	**bpvf**
S-CPCA (non-uniform q-bit)	5	33,27,30,32,20	0.1740	2.0924
	5	51,44,44,50,31	0.0737	3.3308
	5	73,64,56,72,47	0.0294	5.0713
CPCA [8]	5	10	1.47	2
	5	20	0.5	3.8
	5	40	0.16	7.4
LPCA [43]	10	10	1.22	2.2
	30	20	0.47	4.1
	30	20	0.128	6.8

Table 2.5 shows the KG_{err} value 0.1740 for target $bpvf$ value ≈ 2 for the S-CPCA method, whereas

the KG_{err} values 1.47 and 1.22 are found for CPCA and LPCA methods respectively at the same $bpvf$ value. Similarly, at other $bpvf$ values, the proposed method gives better KG_{err} values than the $CPCA$ and the $LPCA$ methods.

Table 2.6: Comparison of proposed S-CPCA with other methods on "Chicken" sequence

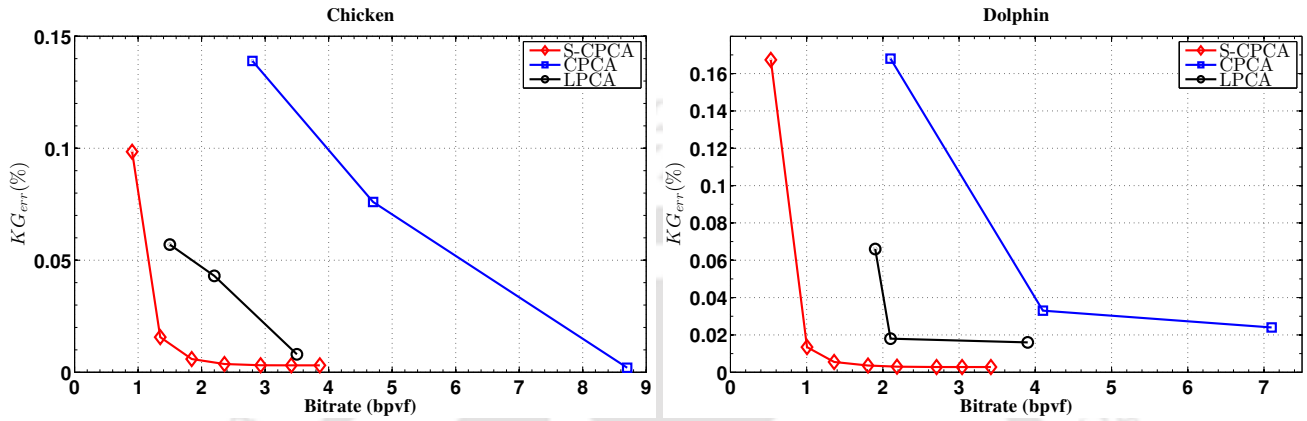
Method	"Chicken" Sequence			
	K	**c per cluster**	**KG_{err}**	**bpvf**
S-CPCA (non-uniform q-bit)	5	33,25,12,24,27	0.0059	1.8439
	5	50,42,25,48,48	0.0031	2.9296
	5	81,73,45,86,85	0.0031	3.8622
CPCA [8]	5	20	0.139	2.8
	10	20	0.076	4.7
	2	60	0.002	8.7
LPCA [43]	10	10	0.057	1.5
	20	10	0.043	2.2
	20	20	0.008	3.5

Table 2.7: Comparison of proposed S-CPCA with other methods on "Dolphin" sequence

Method	"Dolphin" Sequence			
	K	**c per cluster**	**KG_err**	**bpvf**
S-CPCA (non-uniform q-bit)	5	16,16,16,13,21	0.0036	1.8053
	5	30,30,31,22,38	0.0028	2.7045
	5	50,50,51,37,63	0.0028	3.2907
CPCA [8]	2	10	0.168	2.1
	4	10	0.033	4.1
	2	20	0.024	7.1
LPCA [43]	10	5	0.066	1.9
	20	5	0.018	2.1
	20	10	0.016	3.9

Tables 2.6 and 2.7, for the "Chicken" and "Dolphin" sequences respectively, show the minimum KG_{err} values for the proposed method. For all the $bpvf$ values, ≈ 1.8439, ≈ 2.9296 and ≈ 3.8622, the S-CPCA method yields minimum KG_{err} values from 0.0031 to 0.0059 for the "Chicken" sequence. Similarly, for the "Dophin" sequence, the KG_{err} values are found to be minimum for the S-CPCA method. For the CPCA method, a KG_{err} value of 0.002 is found at a much higher $bpvf$ value of 8.7 for the "Chicken" sequence. These results establish that the proposed S-CPCA method is superior to the CPCA and the LPCA methods in terms of compression performance. The RD curves comparing

the S-CPCA method with CPCA and LPCA methods in terms of KG_{err} and $bpvf$ values for the "Chicken" and "Dolphin" sequences are also shown in Figure 2.6. These plots show the improved compression performance of the proposed S-CPCA method.



(a) *Chicken* (b) *Dolphin*

Figure 2.6: Comparison of RD performances of the proposed S-CPCA algorithm with the CPCA and the LPCA algorithms using the KG_{err} metric on 3D animations: (a) *"Chicken"* sequence and (b) *"Dolphin"* sequence

.

2.6 Conclusions

This chapter presented a novel geometry compression algorithm for 3D animations, which outperforms the existing algorithms that use clustering based PCA algorithm. The proposed method provides stable initialization of cluster centres for the K-means clustering algorithm instead of random initialization. The stable cluster centres are obtained by applying a density function based subtractive clustering algorithm on the vertex trajectories. The PCA is applied on each cluster to represent it with a reduced number of PCs, the corresponding PC transformed coefficients and the mean trajectory vector. A non-uniform bit-allocation method is also applied to quantize the elements of PCs and the PC coefficients per cluster. A block based arithmetic coding is proposed to encode the quantization levels of the PCs and PC coefficients quantized with same bits in a cluster. The compression performance of the proposed S-CPCA algorithm is evaluated in terms of KG_{err} and $STED$ metrics for different bitrates measured in terms of the $bpbf$ and compared with other clustered based PCA methods existing in literature. The proposed S-CPCA method performs better than other PCA based methods.

3

Spatio-temporally Scalable Compression of Animation Geometry using SVD

Contents

3.1 Introduction

This chapter investigates the compression of 3D animation geometry in a scalable manner. Scalability is emerging as an important issue in the compression of the 3D animation geometry because of the requirement of visualizing the same animation with different dimensions and qualities in heterogeneous networks and in different end-user devices. A scalable geometry compression algorithm encodes the time varying geometry data in a single bit-stream with different sub-layers and decodes the bit-stream in layers to view the full animation in multiple scales. The decoded animations will have varying frame rates, number of vertices per frame or visual qualities in each layer. The above attributes get enhanced with the decoding of subsequent upper layers.

This chapter proposes an encoder and a decoder structure for achieving the spatio-temporally scalable compression of the 3D animation geometry. The proposed structures use the singular value decomposition (SVD) as the primary compression tool. The elements of the spatial and temporal singular vectors are decomposed into different spatio-temporal layers to get scalable bit-stream. The encoded bit-stream is arranged in a layered structure to achieve temporal and spatial scalability. The connectivity data of the animation is assumed to be isomorphic across the frames and encoded at the beginning of the encoding process using the existing TFAN static mesh compression technique discussed in Section 1.3.6. The proposed scalable encoder and decoder structures are tested on standard 3D animation sequences. The experimental results show good performance in terms of scalable rates and distortions.

The rest of the chapter is organized as follows: Section 3.2 discusses the existing literatures on the scalable compression of the 3D animation geometry. The various approaches for achieving the temporal, spatial and quality scalable compression of animation geometry are outlined. In section 3.3, a compression method using the SVD on the trajectory-based representation of animation geometry matrix has been presented. In section 3.4, the encoder and the decoder structures are presented for the proposed spatio-temporally scalable geometry compression method. The experimental results are discussed in section 3.5. We conclude the chapter in Section 3.6.

3.2 Scalable compression of isomorphic 3D animations - a review

Most of the 3D animation geometry compression algorithms reported in the literature are based on non-scalable or single-rate compression approach. However, there are a few approaches that present the scalable compression of 3D animations. Based on the methods used in those scalable algorithms, they can be broadly classified into four categories as follows:

i) **Scalability using the wavelet transform**: The multi-resolution property of the discrete wavelet transform (DWT) has been exploited by many researchers to provide spatial and temporal scalability in the compression of the 3D animations. Payan and Antonini presented a DWT based compression method [27] for 3D animations, which supports temporal scalability. Lifting based wavelet filtering was applied in the temporal direction to exploit the temporal coherence. A bit allocation scheme [61] was used for sub-bands of wavelet coefficients according to their contribution in the reconstruction quality. The same authors proposed another method for spatio-temporal decomposition of 3D meshes using the *remeshing* technique [41]. The remesher converts an irregular mesh to a semi-regular mesh which is regular in time and also in space. Spatio-temporal decomposition is carried out in the semi-regular meshes. Boulfani-Cuisinaud and Antonini [62] proposed a geometry compression coder combining the temporal wavelet transform (TWT) with a motion-based mesh geometry clustering scheme. The algorithm clusters the input mesh geometry into groups of vertices having the same affine motion parameters and employs a scan-based geometrically compensated temporal wavelet filtering. The wavelet coefficients are quantized and encoded using a bit allocation process and the displacement vectors are entropy encoded. Cho et al. [25] proposed two geometry compression methods for irregular isomorphic 3D animations. In both the methods, the authors applied exact integer spatial wavelet analysis (SWA) to reduce spatial redundancy. For exploiting temporal redundancy, they applied multi-order differential coding (MDC) in one method and the TWT in the other method. The SWA-MDC method offers spatial scalability while the SWA-TWT method provides spatio-temporal multi-resolution coding in a scalable way.

ii) **Scalability using prediction**: Some of the methods have also used geometry based predictive coding schemes for compressing the animations in a spatio-temporally scalable manner.

The predictive coding method exploits the spatial and temporal correlations present among the changing vertex locations of the 3D animations. Stefanoski et al. [38] proposed a patch-based mesh simplification algorithm combined with the linear predictive coding (LPC) approach for spatial scalability. Each frame of a 3D animation is decomposed into spatial layers of vertices from coarse (or base) to fine layers based on the connectivity information. The prediction of the vertex positions in each spatial layer other than the base layer is done by the already encoded 1-ring neighbourhood vertices in the preceding spatial layers. Stefanoski et al. [63] improved the above method by introducing a non-linear predictor along with the linear predictor. The selection between linear and non-linear predictors is decided based on minimum prediction error. In [39], Stefanoski and Ostermann proposed the scalable predictive coding (SPC) algorithm, which uses the LPC in the rotation-invariant space to compensate the local rigid motion of the patches. The layer-wise spatial prediction errors are quantized and then coded using the fast binary arithmetic coding approach [64]. The SPC encoder supports both spatial and temporal scalability with fast encoding/decoding time and low memory requirements for the efficient compression of the 3D animations. The compression performance of the SPC was further improved by Bici and Akar [46, 65] using their novel prediction schemes. The improvement is obtained by using a weighted spatial prediction algorithm which gives unequal weights to the topologically neighbouring vertices during the prediction of vertex locations from previous and current frames. They have also proposed a novel angle based prediction scheme to predict vertex locations in the spatio-temporal layers to obtain better bitrates compared to the SPC. However, it requires more number of operations and higher memory capacity. Both the SPC [39] and the Bici and Akar's algorithm [46] are suitable for low-latency mesh sequences in streaming applications. In [50], Ahn et al. proposed an efficient fine granular scalable coding algorithm for 3D animations, that supports spatial and temporal scalability. The algorithm decimates only a single vertex in each successive spatial layer decomposition to support fine granular scalability and uses both the connectivity and the geometry relationship of vertices to output high quality intermediate resolution meshes. For supporting temporal scalability, the algorithm uses the hierarchical prediction structure of the H.264 video coding [66]. It also applies an efficient bit-plane coding [67] along with the CABAC [12] algorithm to encode the spatial and temporal prediction residuals

for each vertex. The coder is suitable for low latency streaming applications and provides better reduction in bit rates compared to the SPC [39] and the Bici and Akar's prediction methods [46]. The topology data which are used for spatial layer decomposition in the encoder as well as in the decoder are encoded by a low complexity single rate coder TFAN discussed in Section 1.3.6.

iii) **Scalability using skinning technique**: The skinning technique was proposed by Mamou et al. [33] to compress the 3D animations by converting them into a more compact skinning-based representation. In this approach, the mesh vertices are initially segmented into patches consisting of their neighbourhood vertices. The motion of each patch across the frames are described by a set of 3D affine transform parameters calculated for each frame with respect to the first frame. The vertices are then clustered based on their similar affine transform parameters and finally, the motion of each vertex is estimated as a weighted linear combination of the clusters motions. The first frame is compressed using a static mesh encoder proposed by Touma and Gotsman [68]. The residual motion compensation errors for all the mesh vertices in the remaining frames are compressed using a temporal-DCT based encoding scheme. In [42], a method was proposed by combining the skinning based technique with the linear predictive spatially scalable approach of Stefanoski et al. [38] to support both spatially and temporally scalable compression. This method was adopted as a standard by the Moving Picture Experts Group (MPEG) as Amendment 2 of Part 16-AFX (Animation Framework extension) and refereed to as Frame-based Animated Mesh Compression (MPEG-4 FAMC) [69]. This method provides good compression performance and supports three modes of compression: (1) downloadable type with no scalability feature, (2) scalable type supporting both temporal and spatial scalability and (3) streaming type for low latency applications. Mamou et al. [70] proposed two optimizations for the improvement of the MPEG-4 FAMC standard. The first improvement employs the PCA on the motion compensation error residuals. It gives higher compression rates compared to the in-built DCT and DWT compression of residual errors. The second improvement is the optimised model-based adaptive quantization of the subbands of DWT coefficients obtained using the temporal wavelet transform (TWT) of the residual errors.

iv) **Scalability using the SVD**: Heu et al. [16] proposed a method for SNR and temporally

scalable coding of isomorphic 3D animations using the SVD and the bit plane coding. The method applies the SVD on the animation geometry matrix **A** (Equation (1.6)) and uses the spatial and temporal basis vectors to represent the mesh sequence in scalable manner. A few of spatial and temporal basis vectors are selected in decreasing order of their singular values and a bit plane coding algorithm is used to encode the basis vectors. The importance of each bit plane in the basis vectors is calculated as per its contribution to the reconstruction quality of the mesh sequence. The bit planes are transmitted in the decreasing order of their importance to achieve the SNR scalability. To achieve temporal scalability, the elements of the temporal basis vectors are arranged in different group of frames (GOF) and the elements within each GOF are placed in different sub-layers from the base to the higher level layers. A temporal prediction method is applied among the sub-layers of each GOF to predict the elements in the higher levels from the base level. The base layer elements and prediction residual vectors of the higher levels are encoded using bit plane coding and sent to the decoder.

The above literature survey shows that scalable animation geometry compression has its own challenges, particularly in the spatial scalability issues. The following points are noted:

(i) There exist a few approaches for achieving the temporal and spatial scalability in animation geometry compression as discussed above. But none of these works have reported the scalability performance of the methods except the one proposed by Cho et al. [25]. Thus, the scalability performance for the animation geometry compression is not widely explored.

(ii) It is expected that the application of SVD algorithm on the trajectory based geometry matrix **B** (Equation (1.8)) may give better compression results compared to its application on the geometry matrix **A** as used in [16]. This is mainly because of the reduction of SVD computation time and memory requirements. Moreover, the compression method proposed in [16] does not support spatial scalability. It is expected that the layer decomposition (LD) of the elements of spatial basis vectors using the LPC method [38] may facilitate the reconstruction of animation geometry at different spatial layers.

We propose an efficient SVD based spatio-temporally scalable geometry compression algorithm by using the LPC technique on the elements of spatial basis vectors and the hierarchical prediction

structure of the H.264 video coding on the temporal basis vectors.

3.3 Proposed compression of trajectory-based geometry matrix using SVD

The SVD is one of best off-line dimensionality reduction tools for multi-dimensional data. The SVD based non-scalable geometry compression of a 3D animation is obtained by applying it on the frame-based animation geometry matrix $\mathbf{A}$ as reported in [7]. To reduce the computational time and memory requirement, the trajectory-based geometry matrix $\mathbf{B}$ can be decomposed using the SVD. A brief overview with mathematical expression to obtain non-scalable compression using the SVD on matrix $\mathbf{B}$ is presented below.

3.3.1 Geometry compression using SVD on matrix B

Using the SVD, the matrix $\mathbf{B}$ of size $[N_v \times 3N_f]$ is factored into two orthogonal matrices and one diagonal matrix and expressed as

$$\mathbf{B} = \mathbf{U_s}\mathbf{\Sigma_s}\mathbf{W_s}^T \tag{3.1}$$

where $\mathbf{U_s} \in \mathbb{R}^{N_v \times N_v}$ and $\mathbf{W_s} \in \mathbb{R}^{3N_f \times 3N_f}$ are two orthonormal matrices and $\mathbf{\Sigma_s} \in \mathbb{R}^{N_v \times 3N_f}$ is a diagonal matrix. The columns of $\mathbf{U_s}$ and $\mathbf{W_s}$ matrices contain the basis vectors and are expressed as

$$\mathbf{U_s} = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_{N_v} \end{bmatrix} \tag{3.2}$$

and

$$\mathbf{W_s} = \begin{bmatrix} \mathbf{w}_1 & \mathbf{w}_2 & \cdots & \mathbf{w}_{3N_f} \end{bmatrix}, \tag{3.3}$$

where $\mathbf{u}_j \in \mathbb{R}^{N_v}$ and $\mathbf{w}_j \in \mathbb{R}^{3N_f}$ are the j^{th} basis vectors of $\mathbf{U_s}$ and $\mathbf{W_s}$ respectively. The $\mathbf{u}_j$ vectors represent the eigen vectors of the symmetric matrix $\mathbf{B}\mathbf{B}^T$ and $\mathbf{w}_j$ vectors represent the eigen vectors of the symmetric matrix $\mathbf{B}^T\mathbf{B}$. They are respectively called the *left singular vectors* and *right singular vectors* of matrix $\mathbf{B}$. The left singular vectors signify the spatial variations of the geometric components for the N_v vertices and hence, they can be also termed as the *spatial singular vectors*. Similarly, the right singular vectors represent the temporal variations of the geometry for N_f frames with x, y and z component values. Thus, they are termed as the *temporal singular vectors*. The $\mathbf{\Sigma_s}$

matrix can be expressed in the form as given by

$$\mathbf{\Sigma_s} = \begin{bmatrix} \mathbf{\Sigma} & \mathbf{0}_{r\times(3N_f-r)} \\ \mathbf{0}_{(N_v-r)\times r} & \mathbf{0}_{(N_v-r)\times(3N_f-r)} \end{bmatrix}, \tag{3.4}$$

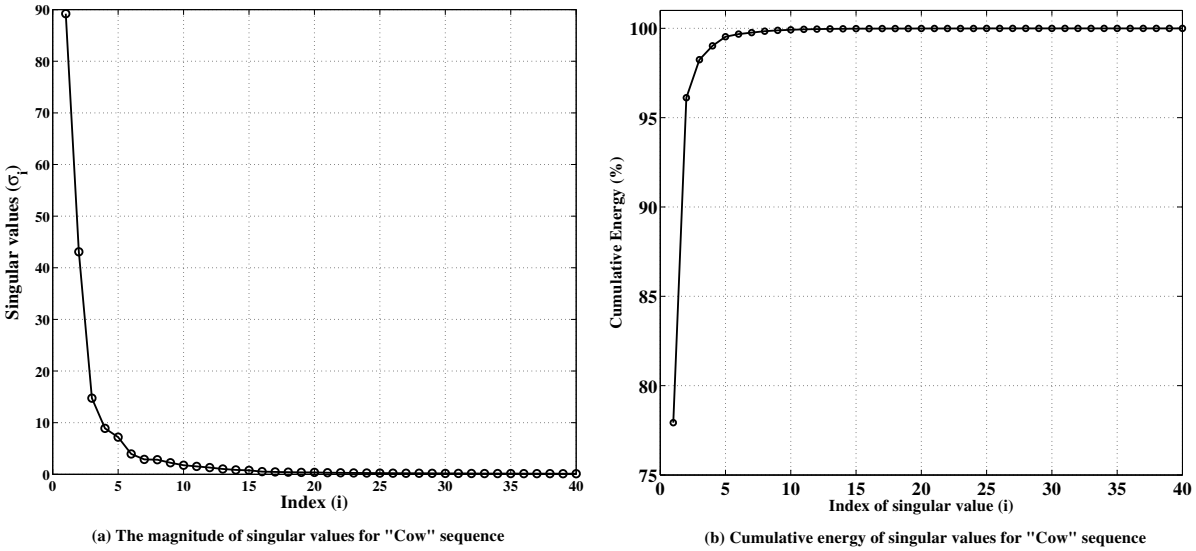
where

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_r \end{bmatrix}_{r\times r} \tag{3.5}$$

and $r = \min\{N_v, 3N_f\}$ represents the maximum rank of the matrix $\mathbf{B}$. The non-negative entries (σ_js) on the main diagonal of $\mathbf{\Sigma_s}$ are called the *singular values* of $\mathbf{B}$. The singular values are ordered in descending order of their magnitudes with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq 0$ and accordingly the singular vectors in $\mathbf{U_s}$ and $\mathbf{W_s}$ are arranged. The r singular values are the square roots of the non-zeros eigen values of both $\mathbf{BB}^T$ and $\mathbf{B}^T\mathbf{B}$. Since, the SVD of matrix $\mathbf{B}$ results in r non-zero singular values, Equation (3.1) can also be represented as the sum of r rank-one matrices obtained from the outer products of the singular vectors of $\mathbf{U_s}$ and $\mathbf{W_s}$ and is given by

$$\mathbf{B} = \sum_{j=1}^{r} \mathbf{B}_j = \sum_{j=1}^{r} \sigma_j \mathbf{u}_j {\mathbf{w}_j}^T \tag{3.6}$$

The magnitude of the singular value σ_j, decreases exponentially with j as illustrated in Figure 3.1(a) by plotting the first 40 singular values of the "Cow" animation sequence with $N_f = 204$ and $N_v = 2904$.



(a) The magnitude of singular values for "Cow" sequence

(b) Cumulative energy of singular values for "Cow" sequence

Figure 3.1: The magnitude of singular values for *"Cow"* sequence

It is observed that the magnitude of σ_1 is the highest with magnitude of 90. So, the energy, ${\sigma_1}^2$, of the signal spanned by the first left and right singular vectors, $\mathbf{u}_1$ and $\mathbf{w}_1$ respectively, is the highest. The signal energy, σ_j^2, is computed from the singular values and the cumulative energy is plotted in Figure 3.1(b). It is observed that most of the signal energy is captured by the first few singular values. Thus it is expected that a good reconstruction of the geometry matrix $\mathbf{B}$ may be possible with a few outer products of singular vectors scaled by the corresponding singular value.

3.3.2 Reconstruction from SVD

The reconstruction of $\mathbf{B}$ after the SVD requires both the spatial and temporal singular vectors and the corresponding singular values. Since the lower order singular values are of negligible energy as shown in Figure 3.1(a), the outer product of $\mathbf{u}_j$ and $\mathbf{w}_j$ is scaled by a very small singular value σ_j for a larger j. Accordingly, the signal energy, σ_j^2, of the scaled outer product $\mathbf{B}_j = \sigma_j \mathbf{u}_j \mathbf{w}_j^T$ also decreases exponentially. Therefore, the most of the energy of the animation geometry matrix $\mathbf{B}$ is concentrated on the first few outer products. A good approximation of reconstructed geometry matrix $\tilde{\mathbf{B}}$ can be obtained by the sum of the first k ($k \ll r$) rank-one matrices. The rank-one matrices are obtained

from the outer products of the first k singular vectors $\mathbf{u}_j$ and $\mathbf{w}_j$. Thus, $\tilde{\mathbf{B}}$ is given by

$$\tilde{\mathbf{B}} = \sum_{j=1}^{k} \sigma_j \mathbf{u}_j \mathbf{w}_j^T = \begin{bmatrix} \mathbf{u_1} & \mathbf{u_2} & \cdots & \mathbf{u_k} \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_k \end{bmatrix} \begin{bmatrix} \mathbf{w_1}^T \\ \mathbf{w_2}^T \\ \vdots \\ \mathbf{w_k}^T \end{bmatrix} \tag{3.7}$$

The application of the SVD on $\mathbf{B}$ facilitates data compression by representing the data using only k singular values and their corresponding spatial and temporal singular vectors. The selection of k may be based on the required reconstruction quality of the animation geometry data. Figure 3.2 shows the reconstructed frame no. 107 of the "Cow" sequence using varying number of singular values and their corresponding spatial and temporal singular vectors.

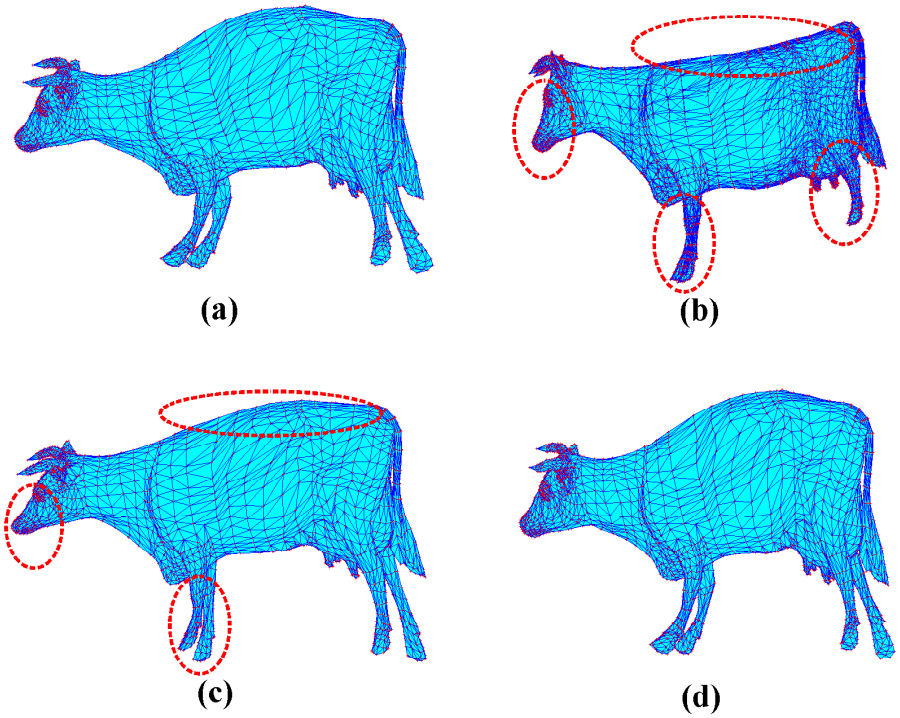


Figure 3.2: Reconstructed frames using varying number of spatial and temporal basis vectors for *"Cow"* sequence (frame no. 107) : (a) original frame, (b) reconstructed frame using 3 basis vectors, (c) reconstructed frame using 10 basis vectors and (d) reconstructed frame using 40 basis vectors.

It is observed that the reconstructed frame using 3 basis vectors, shows distortions at many areas with respect to the original frame shown in Figure 3.2(a). These areas are encircled in Figure 3.2(b).

Similarly, the reconstructed frame using 10 number of basis vectors is shown in Figure 3.2(c). This frame shows comparatively better reconstruction. Significant distortion is still noticed at many areas as marked in the figure. When the number of basis vectors are increased to 40, the reconstructed frame (Figure 3.2(d)) is perceptually almost same as the original frame (Figure 3.2(a)).

3.4 Proposed spatio-temporally scalable animation geometry coder using the SVD

The compression technique using the SVD on the **B** matrix as discussed above is not scalable in nature. For supporting spatial and temporal scalability of the animation geometry, it is required to encode the geometry data into a single bit-stream so that the decoder can reconstruct the data at various scales starting from the base layer to the full enhancement layers. We note the following two properties related to the reconstruction of the geometry data from the SVD on geometry matrix **B**:

(i) By selecting a set of geometric elements from the spatial singular vectors and then multiplying it by the singular values and temporal singular vectors, the geometry data of the corresponding vertices can be reconstructed across all the frames in the spatial domain. In other words, by sampling $\mathbf{u}_j$s, Equation (3.7) can be used to reconstruct all the frames with reduced spatial dimensions.

(ii) By selecting a set of elements from the temporal basis vectors and then multiplying it by the singular values and spatial basis vectors, the geometry data of all the vertices can be reconstructed for the corresponding frames in the spatial domain. In other words, by sampling $\mathbf{v}_j$s, Equation (3.7) can be used to reconstruct all the vertices with reduced number of frames.

The above properties are useful for layer-wise reconstruction of the animation geometry for scalable coding. To establish the above facts, suppose $v_{i_x}^f$, $v_{i_y}^f$ and $v_{i_z}^f$ represent respectively the x, y and z coordinates of the i^{th} vertex of the f^{th} frame, $\mathbf{p}_{v_i}^f$, in the geometry matrix **B**. We also assume that u_i^j denotes the i^{th} element of the j^{th} spatial singular vector $\mathbf{u}_j$ and corresponds to the spatial geometric component for the i^{th} vertex in the SVD domain. Similarly, we assume that $w_{f_x}^j$, $w_{f_y}^j$ and $w_{f_z}^j$ are consecutive three elements of the j^{th} temporal singular vector $\mathbf{w}_j$, where $f = 1, \ldots, N_f$. These consecutive three elements of $\mathbf{w}_j$ vectors correspond to the temporal components in the x, y and z

axes respectively for the f^{th} frame. The elements corresponding to x, y and z temporal components for f-th frame are located at $(3f-2)$, $(3f-1)$ and $3f$ respectively in each $\mathbf{w}_j$ vector. So, we can rewrite Equation (3.6) as

$$\begin{bmatrix} v^1_{1_x} & v^1_{1_y} & v^1_{1_z} & \cdots & v^{Nf}_{1_x} & v^{Nf}_{1_y} & v^{Nf}_{1_z} \\ v^1_{2_x} & v^1_{2_y} & v^1_{2_z} & \cdots & v^{Nf}_{2_x} & v^{Nf}_{2_y} & v^{Nf}_{2_z} \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ v^1_{N_{v_x}} & v^1_{N_{v_y}} & v^1_{N_{v_z}} & \cdots & v^{N_f}_{N_{v_x}} & v^{N_f}_{N_{v_y}} & v^{N_f}_{N_{v_z}} \end{bmatrix} =$$

$$\begin{bmatrix} u^1_1 & u^2_1 & \cdots & u^r_1 \\ u^1_2 & u^2_2 & \cdots & u^r_2 \\ \vdots & \vdots & \ddots & \vdots \\ u^1_{N_v} & u^2_{N_v} & \cdots & u^r_{N_v} \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_r \end{bmatrix} \begin{bmatrix} w^1_{1_x} & w^2_{1_x} & \cdots & w^r_{1_x} \\ w^1_{1_y} & w^2_{1_y} & \cdots & w^r_{1_y} \\ w^1_{1_z} & w^2_{1_z} & \cdots & w^r_{1_z} \\ \vdots & \vdots & \cdots & \vdots \\ w^1_{N_{fx}} & w^2_{N_{fx}} & \cdots & w^r_{N_{fx}} \\ w^1_{N_{fy}} & w^2_{N_{fy}} & \cdots & w^r_{N_{fy}} \\ w^1_{N_{fz}} & w^2_{N_{fz}} & \cdots & w^r_{N_{fz}} \end{bmatrix}^T \tag{3.8}$$

The spatial domain geometry coordinates of the vertex $\mathbf{p}^f_{v_i} = \begin{bmatrix} v^f_{i_x} v^f_{i_y} v^f_{i_z} \end{bmatrix} \in \mathbb{R}^3$ can now be reconstructed using the singular values and the corresponding elements of the $\mathbf{u}_j$ and $\mathbf{w}_j$ vectors. The spatial domain coordinates of the vertex $\mathbf{p}^f_{v_i}$ can be written as

$$\mathbf{p}^f_{v_i} = \begin{bmatrix} v^f_{i_x} \\ v^f_{i_y} \\ v^f_{i_z} \end{bmatrix}^T = \begin{bmatrix} \sum_{j=1}^{r} \sigma_j u^j_i w^j_{f_x} \\ \sum_{j=1}^{r} \sigma_j u^j_i w^j_{f_y} \\ \sum_{j=1}^{r} \sigma_j u^j_i w^j_{f_z} \end{bmatrix} \tag{3.9}$$

The Equation (3.9) can also be written in the matrix form as

$$\mathbf{p}^f_{v_i} = \begin{bmatrix} v^f_{i_x} \\ v^f_{i_y} \\ v^f_{i_z} \end{bmatrix}^T = \begin{bmatrix} u^1_i & u^2_i & \cdots & u^r_i \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_r \end{bmatrix} \begin{bmatrix} w^1_{f_x} & w^2_{f_x} & \cdots & w^r_{f_x} \\ w^1_{f_y} & w^2_{f_y} & \cdots & w^r_{f_y} \\ w^1_{f_z} & w^2_{f_z} & \cdots & w^r_{f_z} \end{bmatrix}^T \tag{3.10}$$

Thus, the spatial domain geometry coordinates of the vertex $\mathbf{p}_{v_i}^f$ for all the N_f frames of the animation, $\begin{bmatrix} \mathbf{p}_{v_i}^1 & \mathbf{p}_{v_i}^2 & \cdots & \mathbf{p}_{v_i}^{N_f} \end{bmatrix}$, can be reconstructed using the i^{th} row element across the r singular vectors $\mathbf{u}_j$, the singular values and all the elements of the $\mathbf{w}_j$ vectors as

$$\begin{bmatrix} \mathbf{p}_{v_i}^1 & \mathbf{p}_{v_i}^2 & \cdots & \mathbf{p}_{v_i}^{N_f} \end{bmatrix} = \begin{bmatrix} u_i^1 & u_i^2 & \cdots & u_i^r \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_r \end{bmatrix} \begin{bmatrix} w_{1_x}^1 & \cdots & w_{1_x}^r \\ w_{1_y}^1 & \cdots & w_{1_y}^r \\ w_{1_z}^1 & \cdots & w_{1_z}^r \\ \vdots & \cdots & \vdots \\ w_{N_{fx}}^1 & \cdots & w_{N_{fx}}^r \\ w_{N_{fy}}^1 & \cdots & w_{N_{fy}}^r \\ w_{N_{fz}}^1 & \cdots & w_{N_{fz}}^r \end{bmatrix}^T \tag{3.11}$$

Let us consider a set of l predefined vertices, $\mathbf{p}_{v_{g_1}}^f, \mathbf{p}_{v_{g_2}}^f, \ldots, \mathbf{p}_{v_{g_l}}^f$, spanning over $f = 1, \ldots, N_f$ frames have indices $g_1, g_2, \ldots, g_l$, where $g_l \in \{1, \ldots, N_v\}$. Let a sub-animation $\mathbf{B_s} \in \mathbb{R}^{l \times 3N_f} \subseteq \mathbf{B}$ with the above l predefined vertices is to be reconstructed using the elements of $\mathbf{u}_j$ and $\mathbf{v}_j$ vectors for $j = 1, \ldots, r$. Let us assume that elements $u_{g_1}^j, u_{g_2}^j, \ldots, u_{g_l}^j$ in the j^{th} spatial singular vector, $\mathbf{u}_j$, represent the corresponding l geometry components. Since l vertices in $\mathbf{B_s}$ will be reconstructed for N_f frames, all the elements of $\mathbf{w}_j$ vectors having x, y and z temporal components for N_f frames will be used for reconstruction. The reconstruction equation is given as

$$\mathbf{B_s} = \underbrace{\begin{bmatrix} u_{g_1}^1 & u_{g_1}^2 & \cdots & u_{g_1}^r \\ u_{g_2}^1 & u_{g_2}^2 & \cdots & u_{g_2}^r \\ \vdots & \vdots & \ddots & \vdots \\ u_{g_1}^1 & u_{g_l}^2 & \cdots & u_{g_l}^r \end{bmatrix}}_{(l \times r)} \underbrace{\begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_r \end{bmatrix}}_{(r \times r)} \underbrace{\begin{bmatrix} w_{1_x}^1 & w_{1_x}^2 & \cdots & w_{1_x}^r \\ w_{1_y}^1 & w_{1_y}^2 & \cdots & w_{1_y}^r \\ w_{1_z}^1 & w_{1_z}^2 & \cdots & w_{1_z}^r \\ \vdots & \vdots & \cdots & \vdots \\ w_{N_{fx}}^1 & w_{N_{fx}}^2 & \cdots & w_{N_{fx}}^r \\ w_{N_{fy}}^1 & w_{N_{fy}}^2 & \cdots & w_{N_{fy}}^r \\ w_{N_{fz}}^1 & w_{N_{fz}}^2 & \cdots & w_{N_{fz}}^r \end{bmatrix}^T}_{(3N_f \times r)} \tag{3.12}$$

The spatial domain geometry data of $\mathbf{B_s}$ can be expressed as

$$\mathbf{B_s} = \begin{bmatrix} v^1_{g_{1,x}} & v^1_{g_{1,y}} & v^1_{g_{1,z}} & \cdots & v^{N_f}_{g_{1,x}} & v^{N_f}_{g_{1,y}} & v^{N_f}_{g_{1,z}} \\ v^1_{g_{2,x}} & v^1_{g_{2,y}} & v^1_{g_{2,z}} & \cdots & v^{N_f}_{g_{2,x}} & v^{N_f}_{g_{2,y}} & v^{N_f}_{g_{2,z}} \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ v^1_{g_{l,x}} & v^1_{g_{l,y}} & v^1_{g_{l,z}} & \cdots & v^{N_f}_{g_{l,x}} & v^{N_f}_{g_{l,y}} & v^{N_f}_{g_{l,z}} \end{bmatrix}_{(l \times 3N_f)} \tag{3.13}$$

The reconstruction of sub-animation represented by $\mathbf{B_s}$ is possible by using l predefined spatial geometric components from r number of $\mathbf{u}_j$ vectors, singular values and all the temporal components of the $\mathbf{w}_j$ vectors. Thus, the original geometry matrix $\mathbf{B}$ can be reconstructed at different layers if the N_v geometric components in $\mathbf{u}_j$ vectors are hierarchically divided into different spatial groups. This will facilitate in the reconstruction of geometry data in a spatially scalable manner. In this work, animations with hierarchical spatial layers are generated by applying a patch based mesh simplification algorithm using the connectivity information. Each spatial layer will have predefined set of vertices such that vertices at the upper spatial layers are predicted using the vertices in the subsequent lower spatial layers. The details of the spatial layer design algorithm is presented in Section 3.4.1.

Similarly, by looking at Equation (3.12), the temporal components in $\mathbf{w}_j$ vectors, which correspond to the frames in spatio-temporal domain, can be divided into different layers to reconstruct the animation geometry at different frame rates. The elements of $\mathbf{w}_j$ vectors can be decomposed into TL number of temporal layers following the hierarchical prediction structure of the H.264 video coding [66]. The $3N_f$ temporal components in each $\mathbf{w}_j$ vector, corresponding to N_f components of each x, y and z coordinates, can be sampled in power of two. This results frame rates of $N_f/2$, $N_f/4$, $N_f/8$ and so on, getting temporal scalability at decoder. We can reconstruct the geometry data of a sub-animation, $\mathbf{B_t} \in \mathbb{R}^{l \times 3t} \subseteq \mathbf{B}$ comprising predefined l vertices and t frames using the elements of $\mathbf{u}_j$ and $\mathbf{v}_j$ vectors for $j = 1, \ldots, r$. Let us assume that $g_1, g_2, \ldots, g_l$, where $g_l \in \{1, \ldots, N_v\}$, are the indices of l predefined vertices and $\{f_1, f_2, \ldots, f_t\}$, where $f_t \in \{1, \ldots, N_f\}$ are the indices of t frames. Let us assume that elements $u^j_{g_1}, u^j_{g_2}, \ldots, u^j_{g_l}$ in the j^{th} spatial singular vector, $\mathbf{u}_j$, represent the corresponding l geometry components. Likewise, let us assume that the elements $\left[\left(w^j_{f_{1,x}}, w^j_{f_{1,y}}, w^j_{f_{1,z}}\right), \left(w^j_{f_{2,x}}, w^j_{f_{2,y}}, w^j_{f_{2,z}}\right), \ldots, \left(w^j_{f_{t,x}}, w^j_{f_{t,y}}, w^j_{f_{t,z}}\right)\right]$ in the j^{th} temporal singular vector, $\mathbf{w}_j$, represent the corresponding temporal geometry components having x, y and z components

for the t number of frames of sub-animation, $\mathbf{B_t}$. The reconstruction equation of $\mathbf{B_t}$ is written as

$$\mathbf{B_t} = \underbrace{\begin{bmatrix} u^1_{g_1} & u^2_{g_1} & \cdots & u^r_{g_1} \\ u^1_{g_2} & u^2_{g_2} & \cdots & u^r_{g_2} \\ \vdots & \vdots & \ddots & \vdots \\ u^1_{g_1} & u^2_{g_l} & \cdots & u^r_{g_l} \end{bmatrix}}_{(l\times r)} \underbrace{\begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_r \end{bmatrix}}_{(r\times r)} \underbrace{\begin{bmatrix} w^1_{f_{1,x}} & w^2_{f_{1,x}} & \cdots & w^r_{f_{1,x}} \\ w^1_{f_{1,y}} & w^2_{f_{1,y}} & \cdots & w^r_{f_{1,y}} \\ w^1_{f_{1,z}} & w^2_{f_{1,z}} & \cdots & w^r_{f_{1,z}} \\ \vdots & \vdots & \cdots & \vdots \\ w^1_{f_{t,x}} & w^2_{f_{t,x}} & \cdots & w^r_{f_{t,x}} \\ w^1_{f_{t,y}} & w^2_{f_{t,y}} & \cdots & w^r_{f_{t,y}} \\ w^1_{f_{t,z}} & w^2_{f_{t,z}} & \cdots & w^r_{f_{t,z}} \end{bmatrix}^T}_{(3t\times r)} \tag{3.14}$$

The spatial domain geometry data of the sub-animation $\mathbf{B_t}$ can be expressed as

$$\mathbf{B}_t = \underbrace{\begin{bmatrix} v^{f_1}_{g_{1,x}} & v^{f_1}_{g_{1,y}} & v^{f_1}_{g_{1,z}} & \cdots & v^{f_t}_{g_{1,x}} & v^{f_t}_{g_{1,y}} & v^{f_t}_{g_{1,z}} \\ v^{f_1}_{g_{2,x}} & v^{f_1}_{g_{2,y}} & v^{f_1}_{g_{2,z}} & \cdots & v^{f_t}_{g_{2,x}} & v^{f_t}_{g_{2,y}} & v^{f_t}_{g_{2,z}} \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ v^{f_1}_{g_{l,x}} & v^{f_1}_{g_{l,y}} & v^{f_1}_{g_{l,z}} & \cdots & v^{f_t}_{g_{l,x}} & v^{f_t}_{g_{l,y}} & v^{f_t}_{g_{l,z}} \end{bmatrix}}_{(l\times 3t)} \tag{3.15}$$

Based on above two principles, we can reconstruct various spatial and temporal layers for the 3D animation data.

3.4.1 Spatio-temporally scalable layer design

In order to reconstruct the input 3D animation geometry in different spatially scalable layers, the elements of the spatial singular vectors are decomposed into hierarchical spatial layers using a patch based layer decomposition (LD) algorithm [38]. After the hierarchical decomposition, prediction of geometric elements between the spatial layers is performed in the SVD domain. Similarly, to reconstruct the animation geometry in form of variable frame rates, the elements of the temporal singular vectors are decomposed in a hierarchical manner as followed in H.264 video coding. The temporal layers follow a hierarchical prediction structure between them. The prediction of elements among the layers is also performed in the SVD domain. The details about the spatio-temporal scalable layer design process are presented below.

Hierarchical spatial layer design using the connectivity data

In the spatial layer decomposition process, the input connectivity data C_d as given by Equation (1.1) are used to structure the vertices in each frame of animation in different spatial groups. The

spatial grouping of vertices is constant across the frames as the input 3D animation is considered to be an isomorphic one. The set of N_v vertex indices, $I_v = \{g_1, g_2, \ldots, g_{N_v}\}$, for each frame is partitioned into S disjoint subsets V_l, $l = S, \ldots, 1$ with connectivity κ_l such that $\bigcup_{l=1}^{S} V_l = I_v$. The LD process starts with $V_S = I_v$ and $\kappa_S = C_d$ at the full enhancement layer and it employs the patch based simplification technique consisting of following two steps:

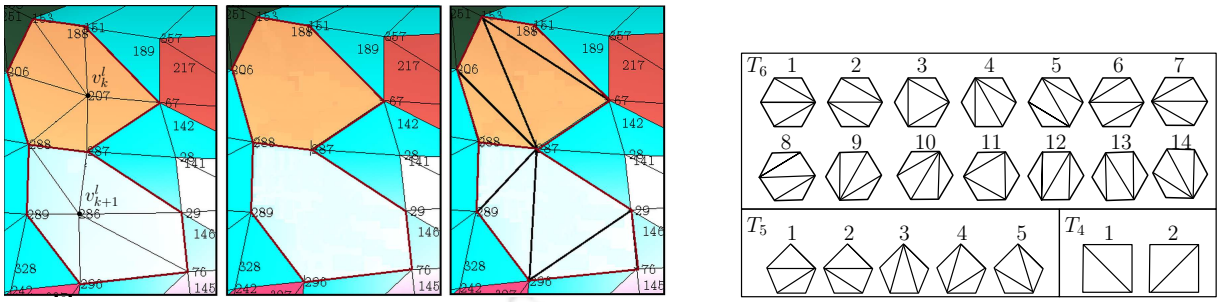
(i) **Patch-based decomposition**: In this step, the degree or valence of each vertex is determined from the connectivity data. After that, a series of K_l non-overlapping patches of valence 6 or lower are determined. A valence-6 patch consists of a set of triangles incident to a vertex of valence 6. Each patch in a layer is described by their middle vertices $g_1^l, \ldots, g_k^l, \ldots, g_{K_l}^l$ as shown in Figure 3.3(a). The set of patches for a layer l is given by

$$R_l = \left\{g_k^l\right\}, \quad k = 1, \ldots, K_l, \quad l = S, \ldots, 1, \tag{3.16}$$

(ii) **Patch simplification**: The patch simplification step involves removing the middle vertex g_k^l of the patches and re-triangulating the remaining polygon as shown in Figure 3.3(a). The number of different triangulations t_d for a degree-d patch depends on the number of its vertices d. The set of all possible triangulations of a degree-d patch is defined as $T_d := \{1, \ldots, t_d\}$ for $d = 4, 5, 6$ as shown in Figure 3.3(b). In order to select a triangulation $\tau \in T_d$ for degree-d patch, a cost function $\Psi\left(g_k^l, \tau\right)$is used to find the average absolute deviation of all valences of the 1-ring neighbourhood vertices of the middle vertex g_k^l from the desired valence 6 and is defined as

$$\Psi\left(g_k^l, \tau\right) := \frac{1}{\left|N\left(g_k^l\right)\right|} \sum_{v \in N\left(g_k^l\right)} \left|\mu\left(v, g_k^l, \tau\right) - 6\right| \tag{3.17}$$

where $N\left(g_k^l\right)$ is the set of 1-ring neighbourhood vertices of the middle patch vertex g_k^l and $\mu\left(v, g_k^l, \tau\right)$ denotes the valence of neighbourhood vertex v after removing the middle vertex g_k^l and triangulating the remaining polygon using any triangulation τ as shown in Figure 3.3(b). In the Equation 3.17, the desired valence is taken as 6 because the average valences of most of vertices for a sufficiently large 3D triangular mesh is found to be 6 as per the Eulers formula [106]. So, the selection of this value reduces the absolute deviation of valences of the vertices from the desired valence 6 after patch decomposition. The triangulation $\tau \in T_d$ which leads to the minimal cost is selected for re-triangulating a patch.



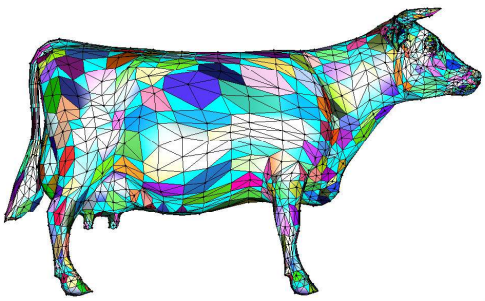
(a) Patch decomposition and simplification **(b)** Triangulations T_d for d = 4, 5, 6.

Figure 3.3: (a) Patch decomposition and simplification, (b) Triangulations T_d for d = 4, 5, 6.

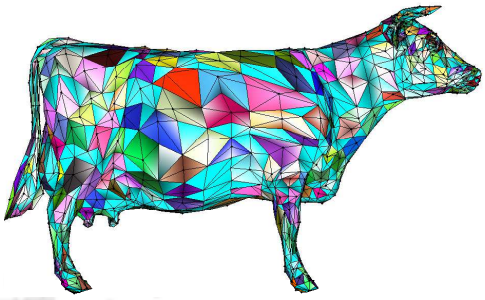
After the above steps, a set of K_S vertices $R_S = \left\{g_1^S, \ldots, g_{K_S}^S\right\}$ is removed and a connected subset of vertices $V_{S-1} = I_v \backslash R_S$ remains with the connectivity data κ_{S-1}. The above patch decomposition and simplification steps are applied iteratively to obtain for each $l = S-1, \ldots, 2$, a set of vertices $R_l = \left\{g_1^l, \ldots, g_{K_l}^l\right\}$ and a simplified connectivity κ_l consisting of vertices $V_{l-1} = I_v \backslash \bigcup_{l=2}^{S} R_l$. The base layer is given by the set of remaining vertices $V_1 = I_v \backslash \bigcup_{l=2}^{S} R_l$. The patch based simplification steps ensures that the neighbourhood of each middle patch vertex $g_k^l \in R_l$ is located in the layer below it i.e $N(g_k^l) \in R_{l-1}$ for $l = 2, \ldots, S$. The spatial layer decompositions upto 8-level for the "Cow" sequence is shown in Figure 3.4. The 8^{th} spatial layer shown in Figure 3.4(a) contain all the 2904 vertices of the "Cow" sequence. The number of vertices in the successive lower spatial layers goes on decreasing as per the patch simplification process as depicted in Figure 3.4(b) to Figure 3.4(h). The base layer contains the lowest number of vertices with $V_1 = 352$. This signifies the possibility of spatially scalable reconstruction of the animation geometry data.

Hierarchical spatial layers of geometric elements in SVD domain

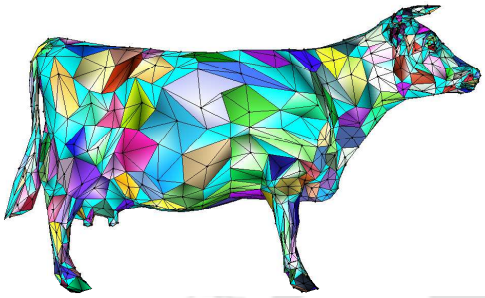
The connectivity based LD algorithm divides the vertices of each frame into non-overlapping hierarchical groups staring from the base layer to the full enhancement layer. This grouping information is used to design the hierarchical spatial layers for geometric elements of the spatial singular vectors in the SVD domain. Let us assume that $u_{i,l}^j \in \mathbb{R}$ represents the i^{th} geometric element of j^{th} spatial singular vector in l-th spatial layer with $u_{i,l}^j \in V_l, \;\; l = 1, \ldots, S, \;\; j = 1, \ldots, k$. Here, k is the number of spatial singular vectors selected after applying SVD on the geometric matrix $\mathbf{B}$. Now, we define



(a) 8^{th} spatial layer, $V_8 = 2904$

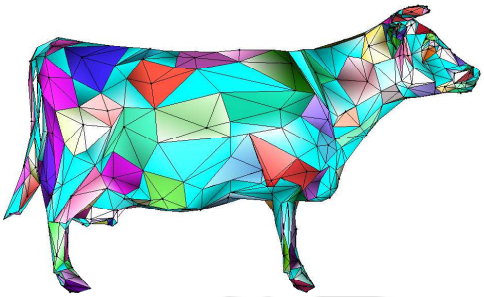


(b) 7^{th} spatial layer, $V_7 = 2145$

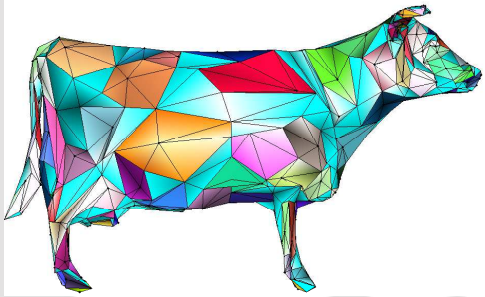


(c) 6^{th} spatial layer, $V_6 = 1564$

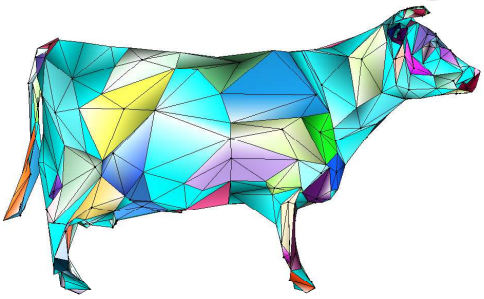
(d) 5^{th} Spatial layer, $V_5 = 1153$



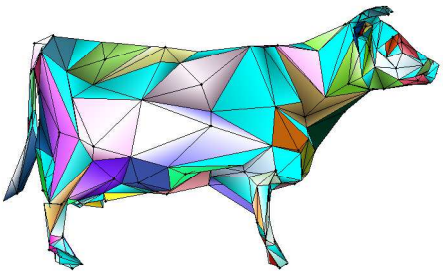
(e) 4^{th} spatial layer, $V_4 = 851$



(f) 3^{rd} Spatial layer, $V_3 = 635$



(g) 2^{nd} spatial layer, $V_2 = 473$



(h) 1^{st} spatial (base) layer, $V_1 = 352$

Figure 3.4: 8-level spatial layer decomposition, "Cow" sequence

each spatial layer in the SVD domain by a matrix $\mathbf{LU}_u^l$ of dimension $|V_l| \times k$ as

$$\mathbf{LU}_u^l = \left[u_{i,l}^j\right] \in \mathbb{R}^{|V_l| \times k} \tag{3.18}$$

Spatial prediction of geometric elements in SVD domain

After the hierarchical decomposition of the elements of the $\mathbf{u}_j$ vectors, spatial prediction of elements between the layers is performed in the SVD domain. The prediction process minimizes the dynamic ranges of the layer-wise geometric elements and helps to reduce the total bits required to encode the elements per layer. The spatial prediction is applied across all the spatial singular vectors to predict the geometric elements of the current layer using their neighbourhood geometric elements present in the lower spatial layer. Spatial prediction is not applied for the geometric elements of the base layer. The geometric elements $u_{i,1}^j \in \mathbf{LU}_u^1$ belonging to the base spatial layer are encoded directly and sent to the decoder. After that, the geometric elements $u_{i,l}^j \in \mathbf{LU}_u^l$ for $l > 1$ of the subsequent higher spatial layers are predicted using the elements of the preceding spatial layer. The prediction of the geometric element $u_{i,l}^j$ for each spatial singular vector is performed by a linear predictor using the average of the set of 1-ring neighbourhood elements, N_{u_i}, of $u_{i,l}^j$ present in the lower spatial layer and given by

$$\tilde{u}_{i,l}^j = \frac{1}{|N_{u_i}|} \sum_{k=1}^{|N_{u_i}|} \tilde{u}_{k,l-1}^j, \quad i \in V_l, \quad N_{u_i} \in V_{l-1}, \quad l = 2, \ldots, S \tag{3.19}$$

The spatial prediction errors for the geometric elements $u_{i,l}^j$ belonging to the higher spatial layers ($l > 1$) are computed as

$$\varepsilon_{u_{i,l}}^j = u_{i,l}^j - \tilde{u}_{i,l}^j, \; l = 2, \ldots, S \; , \; i \in V_l \tag{3.20}$$

Hierarchical temporal layer decomposition in the SVD domain

In the hierarchical temporal layer decomposition, the elements of the temporal singular vectors are divided into different temporal groups. The set of $3N_f$ temporal geometric elements in each $\mathbf{w}_j$ vector, corresponding to N_f components of each x, y and z coordinates, are partitioned into T temporal layers $W^{j,l}$, $l = 1, \ldots, T$. The elements in each $W^{j,l}$ layer correspond to disjoint frame indices with x, y and z coordinates staring from base layer to highest temporal layer such that $\sum_{l=1}^{T} \left|W^{j,l}\right| = 3N_f$. The temporal layer design starts with the sampling of each $\mathbf{w}_j$ vector in the interval of 2^{T-l} to get the set

of temporal geometric elements with their corresponding x, y and z coordinates given as

$$L^{j,l} = \left\{ x : x = \left(\begin{array}{ccc} w^j_{3*2^{T-l}(t-1)+1}, & w^j_{3*2^{T-l}(t-1)+2}, & w^j_{3*2^{T-l}(t-1)+3} \end{array} \right), \; t = 1, 2, \ldots, \frac{N_f}{2^{T-l}} \right\} \tag{3.21}$$

The hierarchical temporal layers $W^{j,l}$ are then formed as

$$\begin{aligned} W^{j,1} &= L^{j,1} \\ W^{j,l} &= L^{j,l} \backslash \bigcup_{k=1}^{l-1} L^{j,k}, \quad l = 2, \ldots, T \end{aligned} \tag{3.22}$$

The layerwise temporal vectors are now obtained by vectorizing the above sets. An example of 4-level decompositions of the elements of a $\mathbf{w}_j$ vector is given as

$$\begin{array}{l} \mathbf{w}^{j,1} = [\ldots (w^j_{(24t-23)}\ w^j_{(24t-22)}\ w^j_{(24t-21)})\ (w^j_{(24t+1)}\ w^j_{(24t+2)}\ w^j_{(24t+3)})\ (w^j_{(24t+25)}\ w^j_{(24t+26)}\ w^j_{(24t+27)}) \ldots]^T \\ \mathbf{w}^{j,2} = [\ldots (w^j_{(24t-11)}\ w^j_{(24t-10)}\ w^j_{(24t-9)})\ (w^j_{(24t+13)}\ w^j_{(24t+14)}\ w^j_{(24t+15)})\ (w^j_{(24t+37)}\ w^j_{(24t+38)}\ w^j_{(24t+39)}) \ldots]^T \\ \mathbf{w}^{j,3} = [\ldots (w^j_{(24t-17)}\ w^j_{(24t-16)}\ w^j_{(24t-15)})\ (w^j_{(24t-5)}\ w^j_{(24t-4)}\ w^j_{(24t-3)})\ (w^j_{(24t+7)}\ w^j_{(24t+8)}\ w^j_{(24t+9)}) \ldots]^T \\ \mathbf{w}^{j,4} = [\ldots (w^j_{(24t-20)}\ w^j_{(24t-19)}\ w^j_{(24t-18)})\ (w^j_{(24t-14)}\ w^j_{(24t-13)}\ w^j_{(24t-12)})\ (w^j_{(24t-8)}\ w^j_{(24t-7)}\ w^j_{(24t-6)}) \ldots]^T \end{array} \tag{3.23}$$

where $\mathbf{w}^{j,l}$ is the vector containing elements for the temporal layer l. We can define each temporal layer in the SVD domain by a matrix of dimension $\left|W^{j,l}\right| \times k$ and is defined as

$$\mathbf{LW}^l_w = \left[w^{j,l}_i \right], \quad i \in W^{j,l}, \;\; l = 1, \ldots, T, \;\; j = 1, \ldots, k \tag{3.24}$$

Temporal prediction of geometric elements in SVD domain

To support temporal scalability, the elements in the successive temporal layers follow a hierarchical prediction pattern excluding the base layer. The temporal elements $W^{j,1}$ belonging to the base temporal layer are encoded directly and sent to the decoder. The elements of the higher temporal layers, $W^{j,l}, l > 1$, are predicted bidirectionally using the elements of the $\mathbf{w}_j$ vectors in the lower level temporal layer. A pictorial representation of the hierarchical prediction of elements in the upper temporal layers is shown in Figure 3.5.

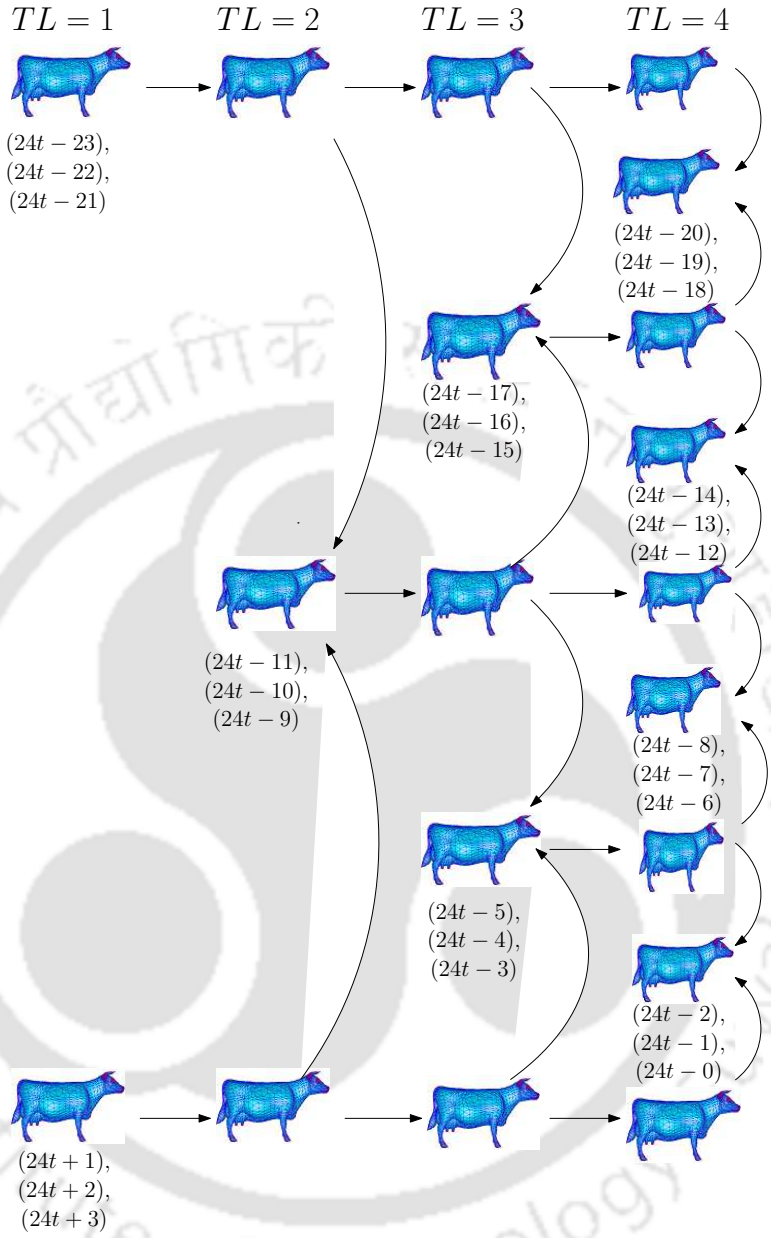


Figure 3.5: Layer-wise prediction of x, y and z temporal elements in $\mathbf{w}_j$ vectors

For computing the temporal prediction errors, for x, y and z temporal components, let us assume that $\varepsilon^j_{w_{i,l}}$ represents the prediction error for the i^{th} element of j^{th} temporal singular vector in l-th temporal layer for the elements in $W^{j,l}$ where $l = 2, \ldots, T$ and $j = 1, \ldots, k$. The prediction error is computed as

$$\varepsilon^j_{w_{i,l}} = w^{j,l}_i - \frac{(w^{j,l-1}_i + w^{j,l-1}_{i+3})}{2}, \qquad i = 1, \cdots, (\left|W^{j,l}\right| - 2) \tag{3.25}$$

3.4.2 Proposed scalable encoder

The proposed encoder structure for the scalable compression of the 3D animation geometry is shown in Figure 3.6. The major steps of proposed encoder are described below.

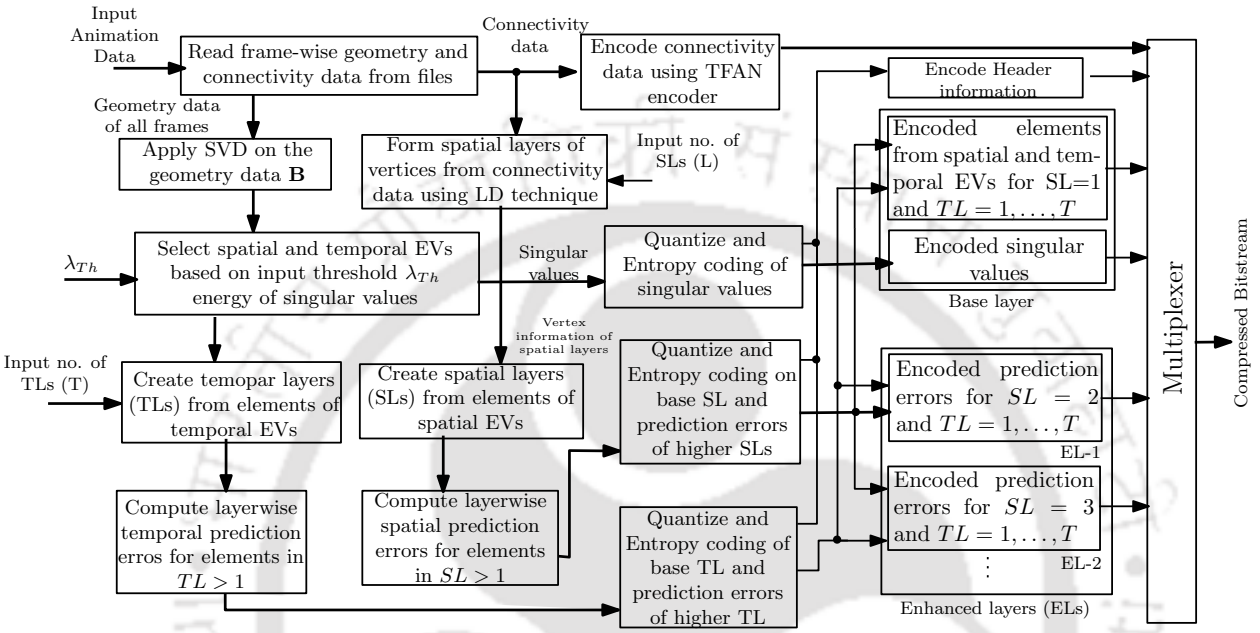


Figure 3.6: Encoder structure for the proposed spatio-temporally scalable compression method

(i) The encoding process starts by reading the files corresponding to frame-wise geometry and connectivity data of the input 3D animation sequence. The geometry data for all the frames are stored in the trajectory based form represented by geometry matrix **B**.

(ii) Apply the SVD on the matrix **B** to reduce the dimension by exploiting the redundancies across the spatio-temporal directions. Select required number of spatial and temporal singular vectors based on a input threshold value λ_{Th}. The λ_{Th} corresponds to the percentage of cumulative energy associated with the singular values.

(iii) Create hierarchical spatial layers by grouping the vertices per frame using the LD technique driven by the connectivity information as described in Section 3.4.1.

(iv) Use the grouping information from LD process in the SVD domain to decompose the elements of the spatial singular vectors from base layer to enhancement layers. The spatial layered decomposition of the elements is consistent across the spatial singular vectors.

(v) Apply spatial prediction across all the spatial singular vectors to predict the geometric elements of the higher layers using their neighbourhood geometric elements present in the lower spatial layer. Spatial prediction is not applied for the geometric elements of the base layer.

(vi) Compute the prediction errors for the elements of the higher spatial layers using Equation 3.20.

(vii) Design the hierarchical temporal layers using the elements of the temporal singular vectors as shown in Equation 3.23. The temporal layers follow a hierarchical prediction structure between them.

(viii) Compute the prediction errors for elements of the enhancing temporal layers using Equation (3.25).

(ix) Allocate a set of non-uniform quantization bits for the geometric elements of spatial and temporal layers based on the singular values using Equation (2.13). The allotted non-uniform quantization bits are same for all the spatial and temporal layers. Encode the sets of non-uniform quantization bits and send it to the decoder.

(x) Quantize the geometric elements belonging to the base spatial and the base temporal layer with the allotted quantization bits. Similarly, quantize the spatial prediction errors, $\varepsilon^{j}_{u_{i,l}}$, and the temporal prediction errors, $\varepsilon^{j}_{w_{i,l}}$, of the enhancement layers with the allotted quantization bits.

(xi) Group the quantization levels of the elements quantized with the same quantization bit for each spatial layer and encode them using the block arithmetic coding described in Section 2.4 to gain compression efficiency.

(xii) Group the quantization levels of the elements of each temporal layer which are quantized with the same quantization bit and encode them using the block arithmetic coding.

(xiii) The singular values are quantized uniformly and entropy coded using an arithmetic encoder and sent along with the base layer.

(xiv) Arrange the encoded bits of all the spatial and temporal layers in a scalable bit pattern from base to enhancement layers before sending it to the decoder.

(xv) Encode the connectivity data using the low complexity TFAN encoder and send it to the decoder.

(xvi) Encode the number of spatial and temporal layers supported in the scalable bit-stream, parameters related to the quantization process as the header information.

3.4.3 Proposed scalable decoder

In the decoder, the reconstruction of the various spatial and temporal layers are done to visualize the 3D animation in a spatio-temporally scalable manner from the coarse to fine. The decoder structure of the proposed scalable 3D animation geometry method is shown in Figure 3.7.

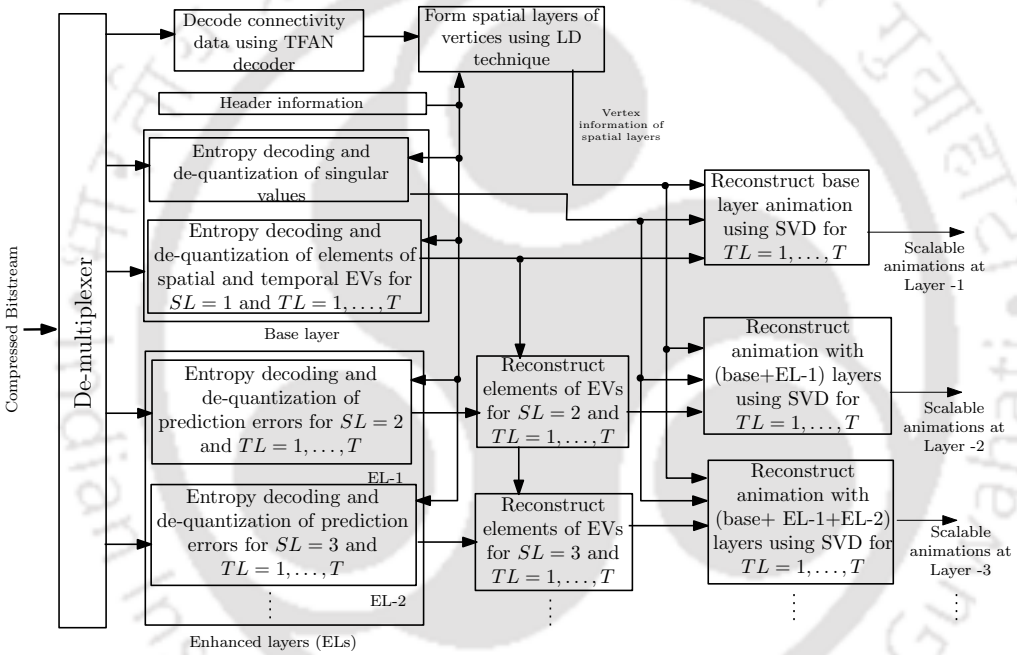


Figure 3.7: Decoder structure of the proposed spatio-temporal scalable method

At the decoder, the following steps are performed:

(i) Decode the header data to extract the information related to the number of spatial and temporal layers and the quantization parameters required for the de-quantization process.

(ii) Separate the encoded connectivity data, singular values, elements of spatial and temporal singular vectors at the base and enhancement layers using a demultiplexer.

(iii) Decode the connectivity data using the TFAN decoder.

(iv) Regenerate the spatial layer grouping information of the vertices based on the decoded connectivity data. This step uses the same patch based mesh simplification techniques as used in the encoder side.

(v) Decode the singular values, geometric elements of the spatial and temporal singular vectors of the base layer using the arithmetic decoder. De-quantize the singular values using the uniform quantization bit and de-quantize the geometric elements of the base layer using the non-uniform quantization bits assigned for them.

(vi) Use the spatial layer grouping information of vertices to reconstruct the base layer animation using the de-quantized singular values and the elements of the spatial and temporal singular vectors of the base layer.

(vii) Decode the prediction errors of the elements of spatial and temporal singular vectors for the enhancement layers using arithmetic decoding and de-quantize them using the non-uniform quantization bits assigned for them.

(viii) Reconstruct the elements of the singular vectors for the enhancement spatial and temporal layers using the decoded base layer and the decoded prediction errors.

(ix) Reconstruct the 3D animation geometry data in a spatio-temporal scalable pattern from base (coarse) to top enhancement layers using the decoded singular values, layer-wise elements of the spatial and temporal singular vectors.

3.5 Experimental results

This section presents the performance of the proposed SVD based scalable geometry compression algorithm. The scalable compression performance of the proposed algorithm has been evaluated on four isomorphic 3D animation sequences namely, “Cow”, “Chicken”, “Dance” and “Face”. The details about these 3D animation data sets are described in Table 2.1 in Chapter 2. Before presenting the scalable compression performance, an experiment is performed to show the advantages of applying the SVD on the trajectory-based geometry matrix **B** over the frame-based geometry matrix **A**. The results for the same is presented below.

3.5.1 Compression performance of SVD on matrix B vs A

In this experiment, the compression performance of using the SVD on the geometry matrix **B** is compared to that on the geometry matrix **A**. Three 3D animation sequences namely, "Cow", "Dolphin" and "Chicken", have been considered in the experiment. The performance is evaluated in terms of SVD computation time, compression ratio (CR), bitrate expressed in number of bits per vertex per frame ($bpvf$) and the KG_{err} metric. The SVD computation time is calculated as the average computation time of SVD algorithm over 20 iterations using the Matlab software loaded on a desktop PC. The desktop PC is running Windows 7, 64-bit operating system with 32GB RAM and Intel core-i5 4570 @3.2GHz processor. A threshold value, $\lambda_{Th} = 99.99\%$, is considered corresponding to the cumulative energy of first k singular values of **A** or **B** matrix such that $\frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_{i=1}^{r} \sigma_i^2} \geq \lambda_{TH}$. A quantization value of $q = 12$ bit is taken to uniformly quantize the elements of the spatial and the temporal singular vectors. The CR value is computed as the ratio of size of the original animation data to the size of the compressed data expressed in bits. To compute the size of original animation data, 12 bit quantization value is considered to represent each geometry element in a lossless manner [3]. Table 3.1 compares the compression parameters obtained by applying the SVD on **B** and **A**.

Table 3.1: Comparison of the compression performances of the SVD on **B** and **A** geometry matrices

	SVD on matrix **B**					SVD on matrix **A**				
Sequence	SVD Time	CR	No. of σ_i	$bpvf$	KG_{err}	SVD Time	CR	No. of σ_i	$bpvf$	KG_{err}
"Cow"	0.355	25.267	20	0.093	1.320	0.975	5.695	35	0.203	1.329
"Chicken"	1.295	95.486	9	0.057	1.590	3.332	34.827	11	0.109	1.586
"Dolphin"	0.900	41.256	7	0.167	0.537	49.265	20.089	5	0.401	0.838

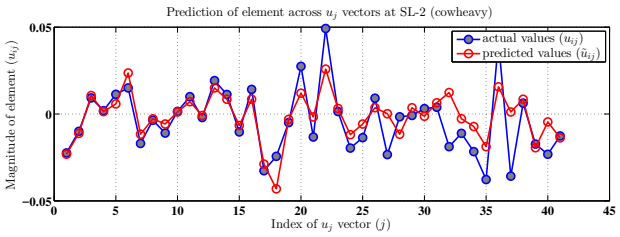
From the results, it is observed that for the "Cow" animation sequence with $N_v = 2904$ and $N_f = 204$, the threshold value of $\lambda_{Th} = 99.99\%$ selects 20 singular values for the SVD on **B** compared to 35 singular values for the SVD on **A**. This gives CR values of 25.267 and 5.695 with reference to the size of original geometry in **B** and **A** matrices respectively. The respective $bpvf$ values are 0.093 and 0.203 for the $KG_{err} \approx 1.32$. The computation time for SVD of **B** is 0.355 seconds compared to 0.975 seconds for SVD of **A**. Similarly, for the "Chicken" and "Dolphin" animation sequences, the SVD of matrix **B** provides better performances in terms of SVD computation time, CR, $bpvf$ and KG_{err} as shown in Table 3.1. Moreover, the total memory requirement for storing the singular vectors

for SVD of **B** is $(N_v \times N_v) + (3N_f \times 3N_f) + 3N_f$ whereas for SVD on **A** the memory requirement is $(3N_v \times 3N_v) + (N_f \times N_f) + N_f$. So, for the animation sequences with $N_v \gg N_f$, the SVD of **B** matrix will require less memory compared to SVD of **A**. Thus, the experimental results justify the selection of matrix **B** over matrix **A** to get better compression performances for the proposed scalable compression algorithm.

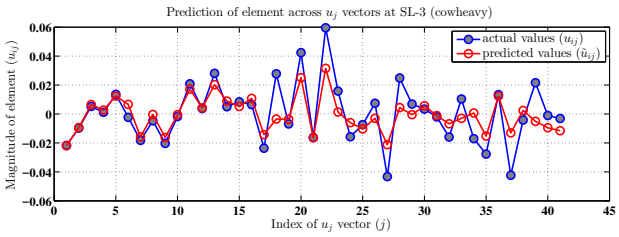
The following sections present the compression performance of the proposed SVD based scalable geometry compression method.

3.5.2 Prediction performances of scalable spatial layers

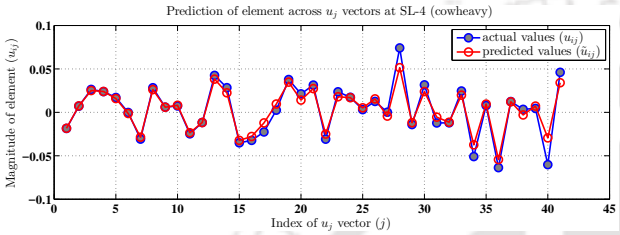
The spatial scalability in the form of varying spatial resolutions (i.e., number of vertices) has been achieved by decomposing the elements of the spatial singular vectors in different layers in a hierarchical manner. The prediction of the elements in the enhancement spatial layers is computed as given in Equation (3.19). The plots for original and predicted values for a few geometric elements present in different spatial layers for the "Cow" sequence is shown in Figure 3.8. From the figure, it is observed that predicted values of the geometric elements $\tilde{u}_i^j$ are close to the actual values u_i^j across the singular vector in all the enhancement spatial layers. As a result, the prediction errors tend to have smaller dynamic range than the original geometric elements. Thus, we can improve the compression performance by allotting a smaller number of bits for the prediction errors. From Figure 3.8(a) and 3.8(b), it is observed that the predicted elements are close to the original values. However, for higher j, the differences between the original and the predicted values of the elements are comparatively higher. This may be due to the widely spaced neighbourhood elements at these lower layers, that affects the averaging process used during the prediction step in the algorithm. However, these high prediction errors for higher j are weighted by smaller singular values during the geometry reconstruction of these spatial layers. So, the contributions of elements for higher j are negligible. Moreover, for elements in the enhancement spatial layers, the differences between the original and predicted values are small as shown in Figures 3.8(e) to 3.8(g). This is because of the availability of a higher number of closely spaced neighbourhood elements that are involved in the prediction process. Similar prediction performances have been observed for the "Chicken", "Dance" and "Face" sequences.



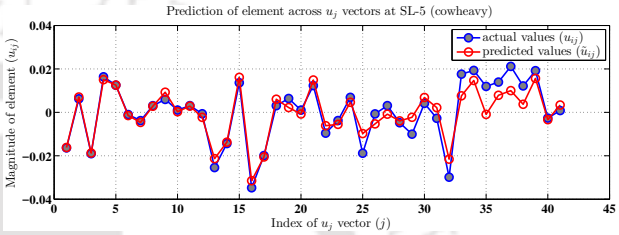
(a) Prediction of geometric elements, $i = 122, SL = 2$



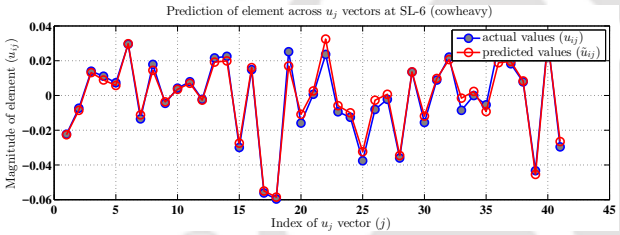
(b) Prediction of geometric elements, $i = 124, SL = 3$



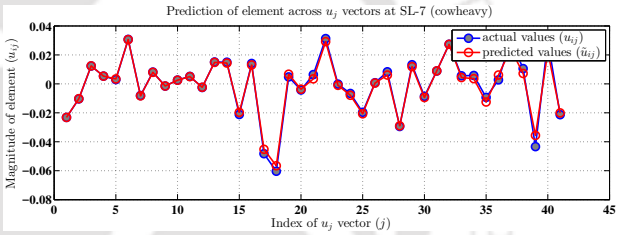
(c) Prediction of geometric elements, $i = 26, SL = 4$



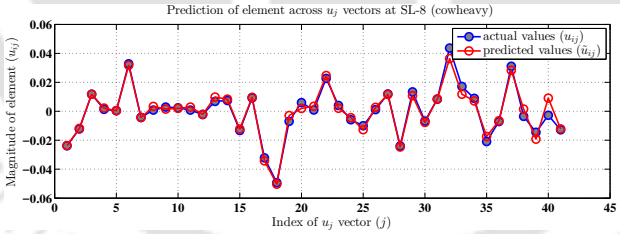
(d) Prediction of geometric elements, $i = 28, SL = 5$



(e) Prediction of geometric elements, $i = 7, SL = 6$



(f) Prediction of geometric elements, $i = 3, SL = 7$



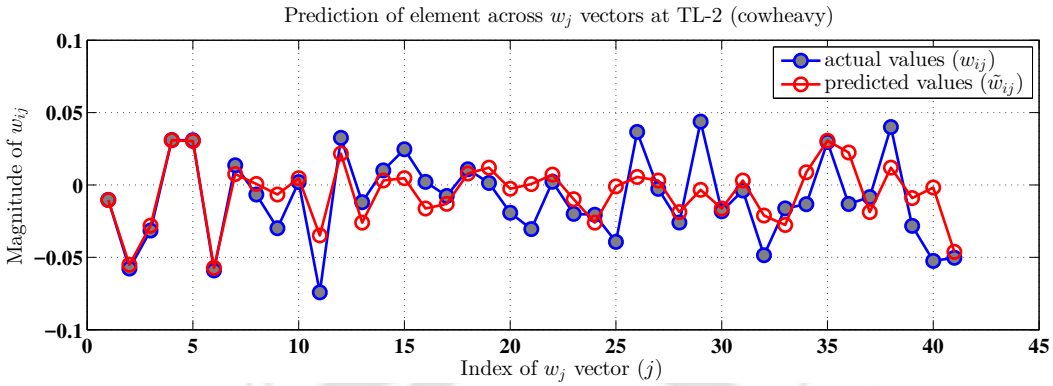
(g) Prediction of geometric elements, $i = 1, SL = 8$

Figure 3.8: Original vs predicted geometric elements at different spatial layers for "Cow" sequence.

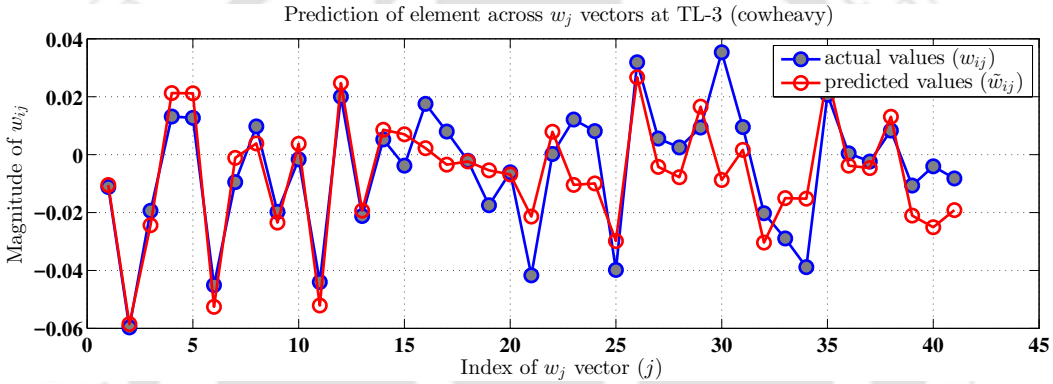
3.5.3 Prediction performances of scalable temporal layers

The temporal scalability in the form of variable frame rates are achieved if the elements of the temporal singular vectors are divided in a hierarchical manner. The elements of the enhancement temporal layers, are predicted bidirectionally using the elements in the lower level temporal layer. The plots for the original and the predicted values for a few temporal elements present in four different

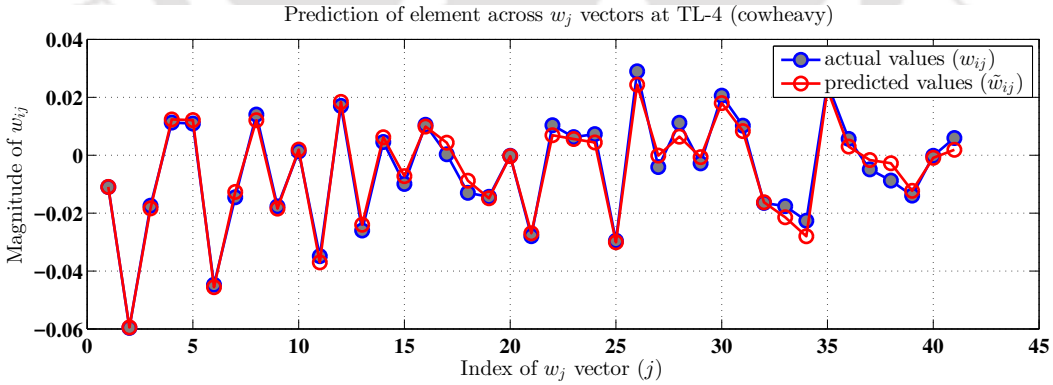
temporal layers for the "Cow" sequence are shown in Figure 3.9. From Figures 3.9(a) and 3.9(b),



(a) Prediction of temporal elements, x-component of 5th frame at $TL = 2$



(b) Prediction of temporal elements, x-component of 3rd frame at $TL = 3$

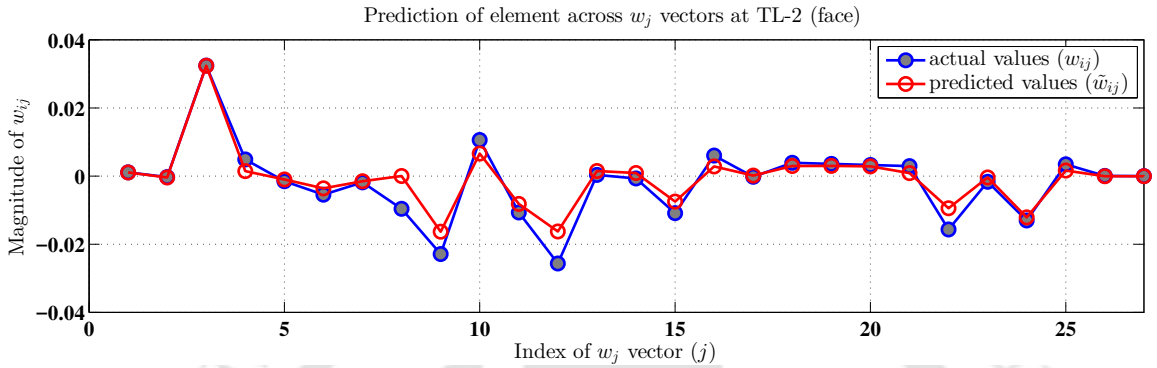


(c) Prediction of temporal elements, x-component of 2nd frame at $TL = 4$

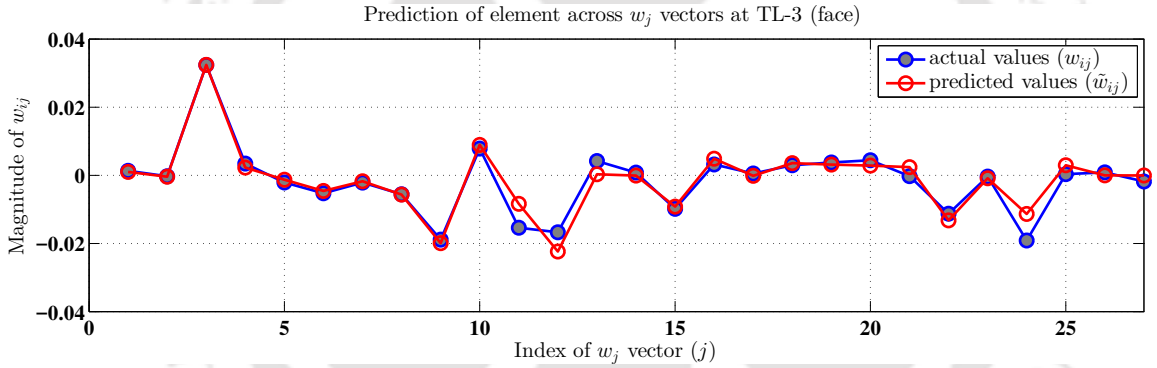
Figure 3.9: Original vs predicted temporal elements at different temporal layers for "Cow" sequence.

it is seen that the differences between the original and predicted values are comparatively higher. The reason may be the widely spaced neighbourhood temporal elements at these layers. The other reason may be the rapid changes of the coordinates across the frames for the "Cow" sequence. In

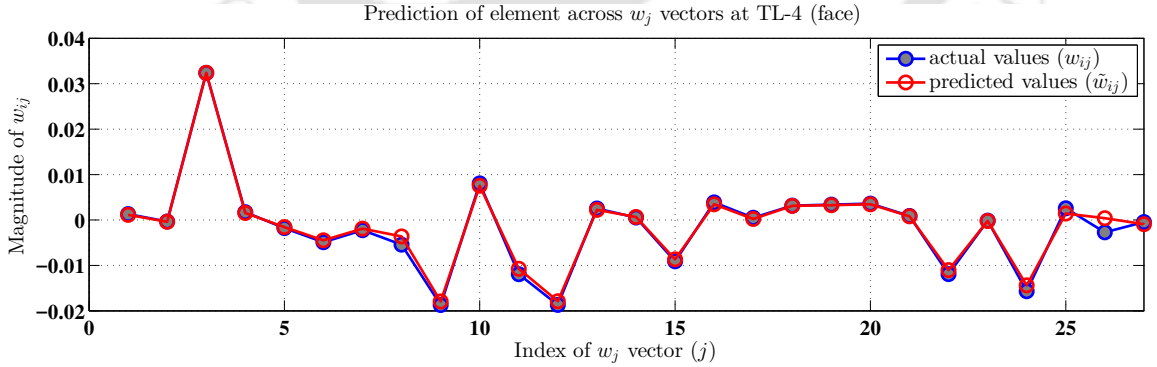
the case of "Face" animation, with smooth changes in motion across the frames, these differences are comparatively lower as shown in Figures 3.10(a) and 3.10(b). Similar prediction performances have been observed for the "Chicken" and "Dance" sequences.



(a) Prediction of temporal elements, x-component of 5th frame at $TL = 2$



(b) Prediction of temporal elements, x-component of 3rd frame at $TL = 3$



(c) Prediction of temporal elements, x-component of 2nd frame at $TL = 4$

Figure 3.10: Original vs Predicted temporal elements at different temporal layers for "Face" sequence.

3.5.4 Scalability performance of the proposed compression method

The scalability performances of the proposed SVD-based spatio-temporally scalable compression method are evaluated on four 3D animations. The performance is evaluated using the KG_{err} and $STED$ metrics for different bitrates expressed in $bpvf$. We have considered up to 8-levels of spatial layer decomposition and 4-levels of temporal layer decomposition. A threshold value $\lambda_{Th} = 99.9\%$ of the cumulative energy of the animation is decided. The first k singular values and the corresponding spatial and temporal singular vectors with $\frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_{i=1}^{r} \sigma_i^2} \geq \lambda_{Th}$ are selected to represent the animation geometry. The k singular values are quantized uniformly using 16 to 18 bits based on an input target bitrate, T_{bpvf}. The elements of spatial and temporal singular vectors at the base layer and prediction errors for the elements at the enhancement spatial and temporal layers are quantized using the non-uniform quantizer presented in Section 2.4 of Chapter 2. The non-uniform quantizer is modified for quantizing the singular vectors of the base layer and the prediction errors of the enhancement layers. These quantized values are finally entropy coded using the arithmetic coding [12] to get the compressed scalable bit-stream. The connectivity data are encoded using the low complexity TFAN encoder. The overhead of scalability has been considered by taking into account the size of compressed bit streams required to compress the connectivity data, geometric elements of base layer with $SL = 1$ and $TL = 1$, spatial and temporal prediction errors of geometric elements in the spatial ($SL > 1$) and temporal ($TL > 1$) enhancement layers, singular values and the header information. The overheads related to computational complexity at the encoder and decoder ends have not been considered in this work. The at the base layer (with SL=1, TL=1) is computed using the encoded file sizes of connectivity data, geometric elements of spatial and temporal singular vectors, singular values and the header information. The $bpvf$ at each enhanced layer (with $SL > 1$ and $TL > 1$) is computed by adding the encoded file size of the preceding spatio-temporal layer with the encoded file size of the prediction errors of the geometric elements for the current layer.

At the decoder, the compressed scalable bit-stream is decoded corresponding to different spatio-temporal layers and the performance metrics are evaluated for layer-wise decompression. The scalability performances on the "Cow" and "Chicken" sequences are evaluated. To compute the KG_{err} and $STED$ metrics for the reconstructed animation at each spatial layer, the original animation data

are arranged in layers to have equal number of vertices at each spatio-temporal layer. The metrics are computed and the values are shown in Tables 3.2 and 3.3 for the "Cow" and "Chicken" sequences respectively.

Table 3.2: Scalability performance in different spatio-temporal layer for "Cow" sequence

"Cow" sequence with $\lambda_{Th} = 99.9\%$, non-uniform quantization, $T_{bpvf} = 1.5$												
layers	TL=1			TL=2			TL=3			TL=4		
	$bpvf$	KG_{err}	$STED$	$bpvf$	KG_{err}	$STED$	$bpvf$	KG_{err}	$STED$	$bpvf$	KG_{err}	$STED$
SL=1	0.2043	0.7385	0.0255	0.2415	0.7584	0.0258	0.3080	0.7709	0.0261	0.4219	0.7749	0.0263
SL=2	0.2566	0.7435	0.0270	0.2937	0.7599	0.0273	0.3603	0.7713	0.0275	0.4742	0.7746	0.0277
SL=3	0.3264	0.7421	0.0279	0.3636	0.7591	0.0282	0.4301	0.7716	0.0285	0.5440	0.7748	0.0287
SL=4	0.4102	0.7419	0.0293	0.4473	0.7580	0.0295	0.5139	0.7696	0.0297	0.6278	0.7729	0.0299
SL=5	0.5278	0.7424	0.0299	0.5650	0.7592	0.0301	0.6315	0.7709	0.0304	0.7454	0.7746	0.0306
SL=6	0.6703	0.7413	0.0303	0.7074	0.7587	0.0305	0.7740	0.7710	0.0308	0.8878	0.7749	0.0310
SL=7	0.8591	0.7429	0.0310	0.8963	0.7591	0.0312	0.9628	0.7717	0.0315	1.0767	0.7755	0.0317
SL=8	1.0930	0.7407	0.0320	1.1301	0.7562	0.0322	1.1967	0.7693	0.0325	1.3106	0.7730	0.0327

Table 3.3: Scalability performance in different spatio-temporal layer for "Chicken" sequence

"Chicken" sequence with $\lambda_{Th} = 99.9\%$, non-uniform quantization, $T_{bpvf} = 1.5$												
layers	TL=1			TL=2			TL=3			TL=4		
	$bpvf$	KG_{err}	$STED$	$bpvf$	KG_{err}	$STED$	$bpvf$	KG_{err}	$STED$	$bpvf$	KG_{err}	$STED$
SL=1	0.2969	0.0509	0.0172	0.3432	0.0510	0.0173	0.4166	0.0524	0.0174	0.5324	0.0553	0.0176
SL=2	0.3299	0.0514	0.0162	0.3762	0.0512	0.0163	0.4496	0.0526	0.0164	0.5654	0.0555	0.0166
SL=3	0.3801	0.0519	0.0149	0.4263	0.0516	0.0150	0.4998	0.0529	0.0151	0.6156	0.0558	0.0154
SL=4	0.4476	0.0522	0.0136	0.4938	0.0517	0.0138	0.5673	0.0531	0.0139	0.6831	0.0560	0.0142
SL=5	0.5331	0.0525	0.0128	0.5793	0.0519	0.0130	0.6528	0.0534	0.0131	0.7686	0.0563	0.0133
SL=6	0.6488	0.0529	0.0119	0.6950	0.0523	0.0121	0.7685	0.0539	0.0123	0.8843	0.0569	0.0125
SL=7	0.8080	0.0530	0.0116	0.8542	0.0524	0.0118	0.9276	0.0539	0.0120	1.0434	0.0569	0.0122
SL=8	1.0242	0.0532	0.0114	1.0704	0.0526	0.0116	1.1439	0.0541	0.0117	1.2597	0.0572	0.0120

From the results shown in Table 3.2 for the "Cow" sequence, it is observed that reconstruction at the base layer with $SL = 1, TL = 1$ requires $bpvf = 0.2043$. But, this shows the KG_{err} value of 0.7385 and the $STED$ value of 0.0255. The error at the base layer is mainly due to the truncation of singular vectors during reconstruction. With the increasing number of spatial layers, the $bpvf$ value goes on increasing and the KG_{err} values also increase due to quantization of prediction errors in the enhancement layers. At spatio-temporal layer with $SL = 8, TL = 1$, $bpvf$ value is 1.0930 resulting KG_{err} value of 0.7407 and $STED$ value of 0.0320. The errors at the enhancement spatio-temporal

layers are due to quantization of prediction errors at the enhancement layers. When we increase the number of temporal layers to $TL = 4$ at $SL = 4$, the $bpvf$ value is 0.6278. The corresponding KG_{err} value is 0.7729 and $STED$ value is 0.0299. At the highest spatio-temporal resolution with $SL = 8, TL = 4$, the $bpvf$ value is 1.3106 which is close to the target bitrate value of $T_{bpvf} = 1.5$. The KG_{err} and $STED$ values are 0.7730 and 0.0327 respectively. For the "Chicken" sequence, similar metrics are evaluated to show the performance of the proposed method and tabulated in Table 3.3. It is observed that the KG_{err} values follow similar trends as the $bpvf$. The values of KG_{err} and $STED$ at the highest spatio-temporal resolution ($SL = 8, TL = 4$) are 0.0572 and 0.0120 respectively. The $bpvf$ value is 1.2597. The scalability performance of the proposed method is also evaluated for the "Dance" and "Face" sequences and the results are shown in Table 3.4 and Table 3.5 respectively.

Table 3.4: Scalability performance in different spatio-temporal layer for "Dance" sequence

"Dance" sequence with $\lambda_{Th} = 99.9\%$, non-uniform quantization, $T_{bpvf} = 2.5$												
layers	TL=1			TL=2			TL=3			TL=4		
	$bpvf$	KG_{err}	$STED$	$bpvf$	KG_{err}	$STED$	$bpvf$	KG_{err}	$STED$	$bpvf$	KG_{err}	$STED$
SL=1	0.3024	0.0129	0.0004	0.3368	0.0132	0.0004	0.3986	0.0129	0.0004	0.5093	0.0130	0.0004
SL=2	0.3928	0.0124	0.0004	0.4272	0.0127	0.0004	0.4890	0.0124	0.0004	0.5997	0.0125	0.0004
SL=3	0.5068	0.0121	0.0004	0.5413	0.0124	0.0004	0.6030	0.0121	0.0004	0.7137	0.0122	0.0004
SL=4	0.6594	0.0119	0.0005	0.6938	0.0122	0.0005	0.7556	0.0119	0.0005	0.8663	0.0120	0.0005
SL=5	0.8810	0.0117	0.0005	0.9154	0.0120	0.0005	0.9772	0.0117	0.0005	1.0879	0.0117	0.0005
SL=6	1.1586	0.0116	0.0005	1.1930	0.0119	0.0005	1.2548	0.0115	0.0005	1.3655	0.0116	0.0005
SL=7	1.7087	0.0113	0.0005	1.7431	0.0116	0.0005	1.8049	0.0113	0.0005	1.9156	0.0113	0.0005
SL=8	2.2828	0.0112	0.0005	2.3173	0.0115	0.0005	2.3790	0.0112	0.0005	2.4897	0.0112	0.0005

Table 3.5: Scalability performance in different spatio-temporal layer for "Face" sequence

"Face" sequence with $\lambda_{Th} = 99.9\%$, non-uniform quantization, $T_{bpvf} = 1.5$												
layers	TL=1			TL=2			TL=3			TL=4		
	$bpvf$	KG_{err}	$STED$	$bpvf$	KG_{err}	$STED$	$bpvf$	KG_{err}	$STED$	$bpvf$	KG_{err}	$STED$
SL=1	0.2117	0.0102	3.16E-05	0.3580	0.0100	3.18E-05	0.6325	0.0097	3.14E-05	1.1427	0.0097	3.16E-05
SL=2	0.2297	0.0101	3.03E-05	0.3759	0.0099	3.05E-05	0.6504	0.0096	3.01E-05	1.1607	0.0096	3.02E-05
SL=3	0.2532	0.0100	2.86E-05	0.3995	0.0098	2.90E-05	0.6740	0.0095	2.85E-05	1.1842	0.0096	2.87E-05
SL=4	0.2809	0.0099	2.90E-05	0.4271	0.0096	2.94E-05	0.7016	0.0093	2.88E-05	1.2119	0.0094	2.91E-05
SL=5	0.3142	0.0097	2.93E-05	0.4605	0.0095	2.98E-05	0.7350	0.0092	2.91E-05	1.2452	0.0092	2.93E-05
SL=6	0.3561	0.0097	3.00E-05	0.5023	0.0094	3.04E-05	0.7769	0.0091	2.98E-05	1.2871	0.0092	3.00E-05
SL=7	0.4125	0.0095	3.01E-05	0.5587	0.0092	3.06E-05	0.8332	0.0090	2.99E-05	1.3435	0.0090	3.02E-05
SL=8	0.4797	0.0095	3.01E-05	0.6259	0.0092	3.04E-05	0.9004	0.0089	2.98E-05	1.4107	0.0090	3.00E-05

The graph showing scalable compression performance of the "Dance" animation for different spatial layers ($SL = 1$ to 8) at a fixed temporal resolution ($TL = 1$) is shown in Figure 3.11(a). It has been observed that the values of the KG_{err} metric decrease with the addition of succeeding spatial enhancement layers and with increase in $bpvf$ values. Similar scalable compression performances have been observed for different spatio-temporal layers from SL=1,TL=1 to SL=4,TL=4 as shown in Figure 3.11(b). The visual performance of the proposed method is shown by plotting an original frame and

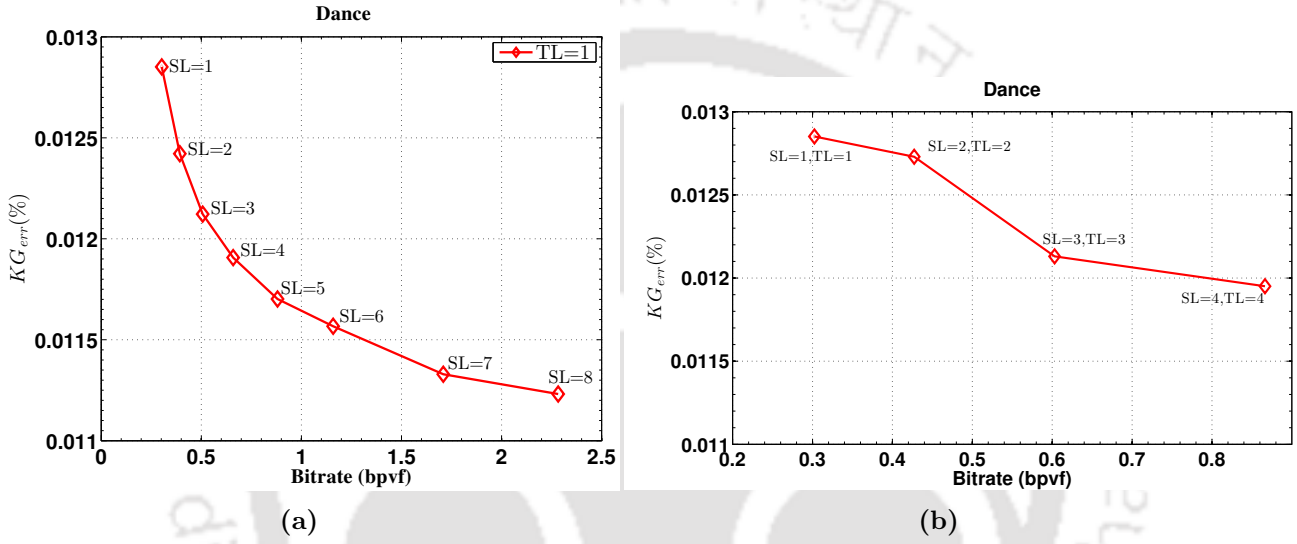
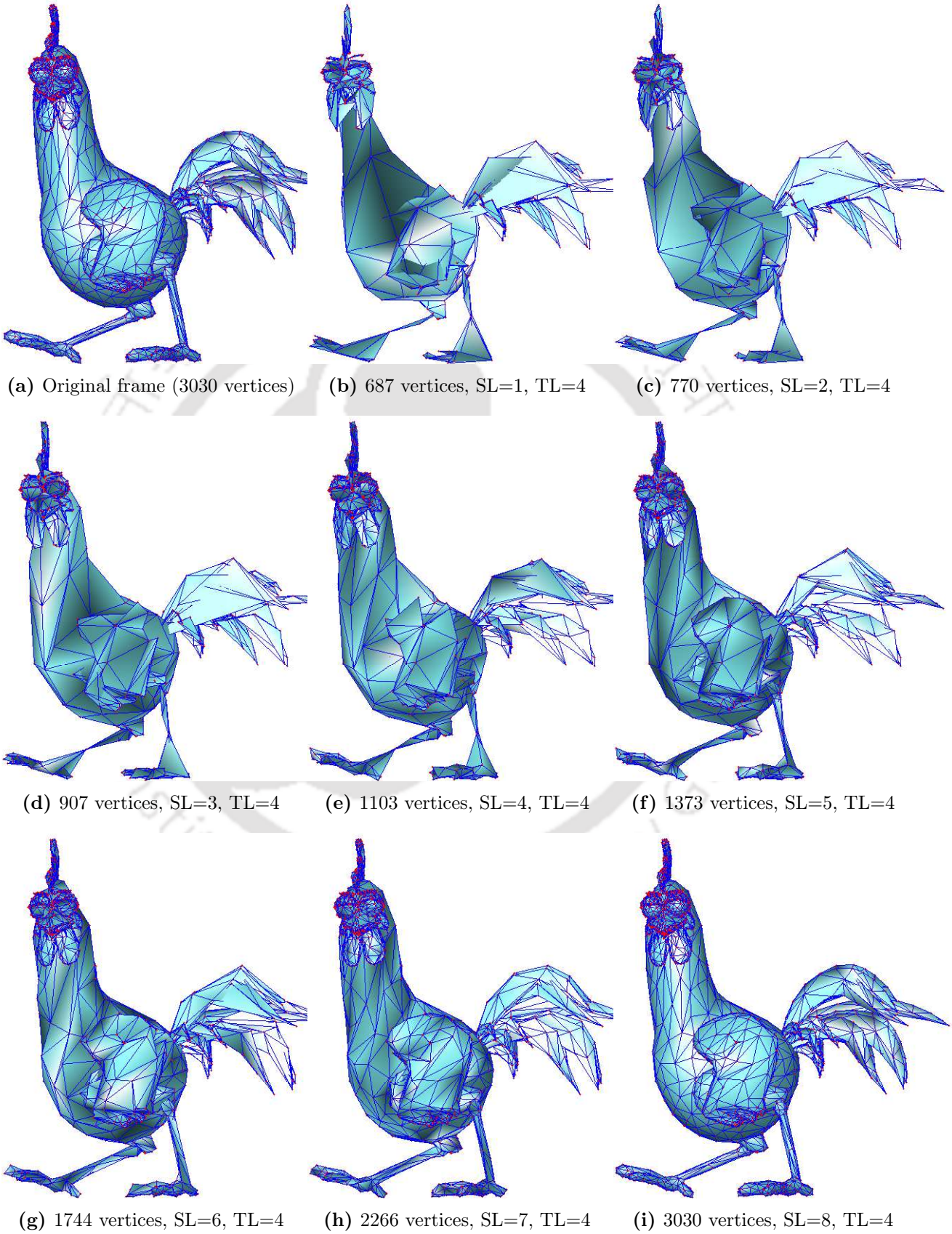


Figure 3.11: Layerwise RD performance of the proposed scalable method using the KG_{err} metric for "Dance sequence at (a) TL=1, SL=1 to 8 and (b) at SL=1,TL=1 to SL=4,TL=4.

the corresponding layer-wise reconstructed frames for an input target bitrate T_{bpvf}. Figures 3.12, 3.13 and 3.14 show the visual performances for the "Chicken", "Dance" and "Face" sequences respectively. Thus, the proposed method shows good spatio-temporally scalable compression performance.

3.5.5 Performance comparison with existing methods

The scalable compression performance of the proposed method is compared with those of FAMC [2], SPC [39] and "Bici & Akar" [46] algorithms. These algorithms have been considered, as they also applied the same patch based mesh simplification techniques to obtain the spatial scalability in different layers. The RD performances have been evaluated in terms of $bpvf$ vs KG_{err} metrics on four 3D animation sequences namely, "Cow", "Chicken", "Dance" and "Face". The RD graphs for the four animations are shown in Figure 3.15. The RD graphs have been generated by plotting the KG_{err}



(a) Original frame (3030 vertices) **(b)** 687 vertices, SL=1, TL=4 **(c)** 770 vertices, SL=2, TL=4

(d) 907 vertices, SL=3, TL=4 **(e)** 1103 vertices, SL=4, TL=4 **(f)** 1373 vertices, SL=5, TL=4

(g) 1744 vertices, SL=6, TL=4 **(h)** 2266 vertices, SL=7, TL=4 **(i)** 3030 vertices, SL=8, TL=4

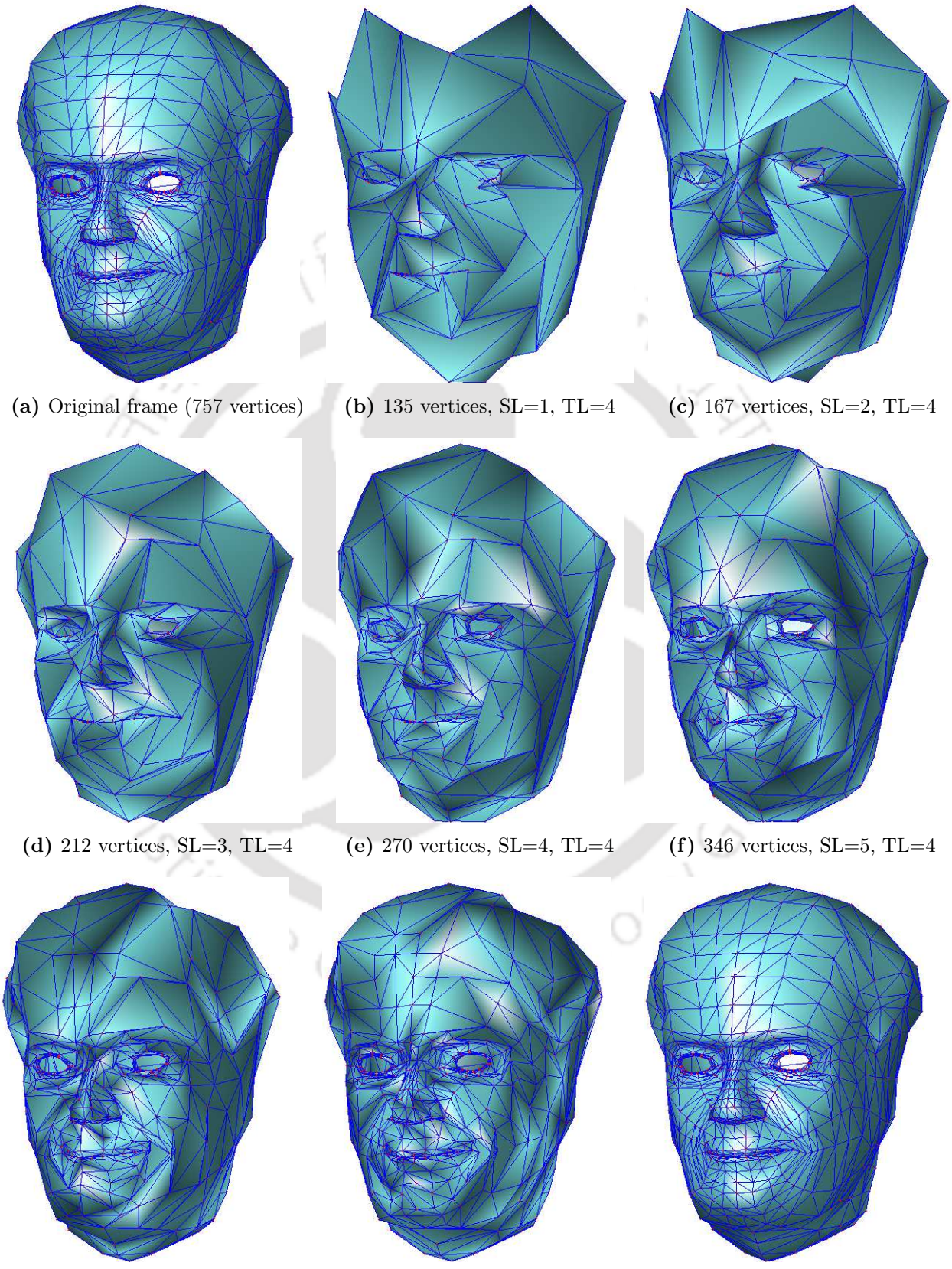
Figure 3.12: Original and reconstructed frames at different spatial layers, "Chicken" sequence, frame no. 163

(a) Original frame (7061 vertices) **(b)** 718 vertices, SL=1, TL=4 **(c)** 958 vertices, SL=2, TL=4

(d) 1275 vertices, SL=3, TL=4 **(e)** 1715 vertices, SL=4, TL=4 **(f)** 2382 vertices, SL=5, TL=4

(g) 3244 vertices, SL=6, TL=4 **(h)** 5006 vertices, SL=7, TL=4 **(i)** 7061 vertices, SL=8, TL=4

Figure 3.13: Original and reconstructed frames at different spatial layers, "Dance" sequence, frame no. 120

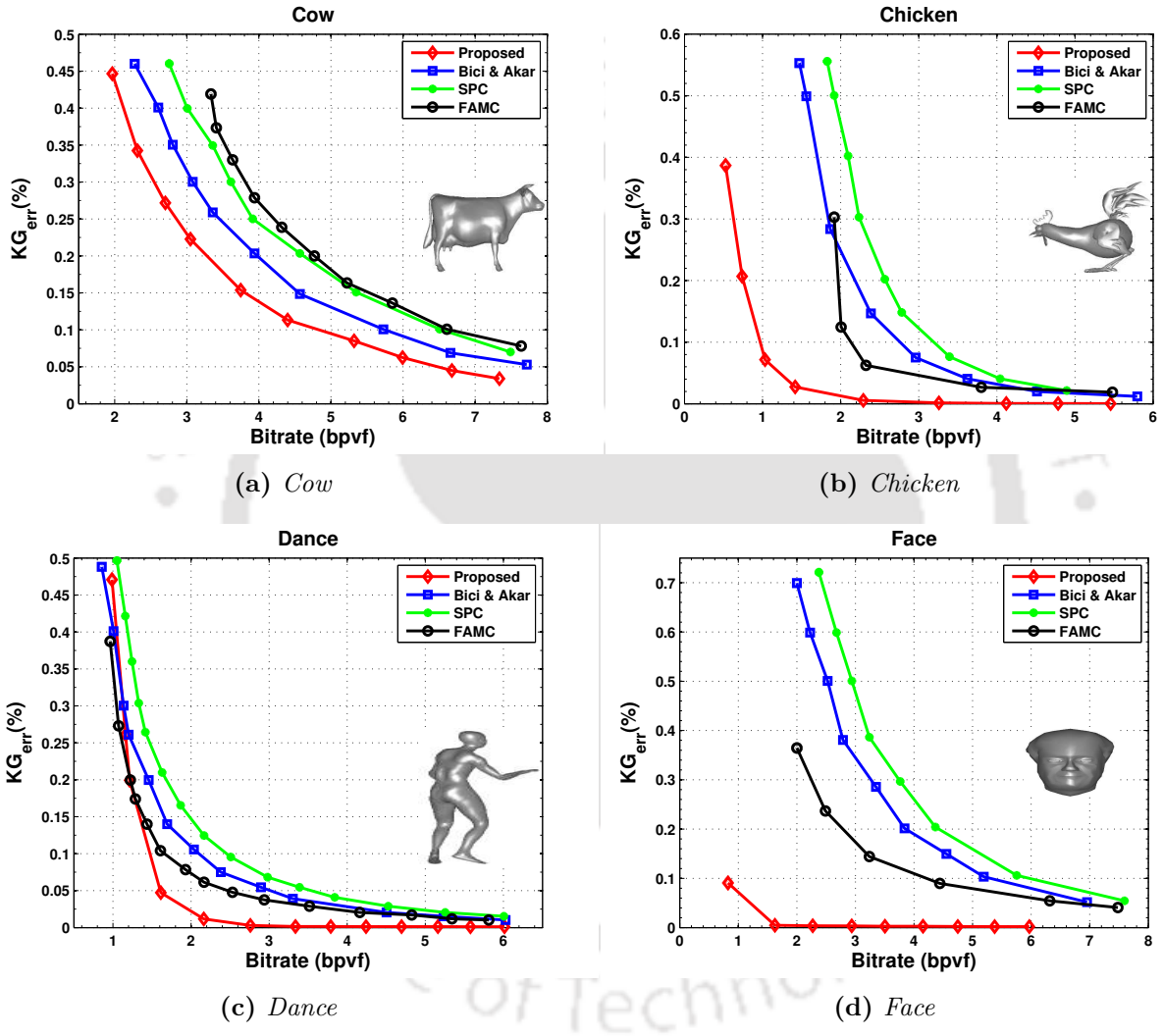


(a) Original frame (757 vertices) **(b)** 135 vertices, SL=1, TL=4 **(c)** 167 vertices, SL=2, TL=4

(d) 212 vertices, SL=3, TL=4 **(e)** 270 vertices, SL=4, TL=4 **(f)** 346 vertices, SL=5, TL=4

(g) 443 vertices, SL=6, TL=4 **(h)** 583 vertices, SL=7, TL=4 **(i)** 757 vertices, SL=8, TL=4

Figure 3.14: Original and reconstructed frames at different spatial layers, "Face" sequence, frame no. 130



(a) *Cow* (b) *Chicken*

(c) *Dance* (d) *Face*

Figure 3.15: RD performance comparison of the proposed algorithm with other scalable methods using the KG_{err} metric for different quantization levels on a few isomorphic 3D animations: (a) *"Cow"* sequence , (b) *"Chicken"* sequence, (c) *"Dance"* sequence and (d) *"Face"* sequence

values obtained at the highest spatio-temporal layer ($SL = 8, TL = 4$) for different T_{bpvf}. The performance of the "Cow" sequence is compared in Figure 3.15(a). It shows better RD performance at all $bpvf$ values compared to FAMC, SPC and "Bici & Akar" algorithms. Similar results are obtained for the "Chicken" and "Face" sequences as shown in Figures 3.15(b) and 3.15(d) respectively. For the "Dance" sequence, at lower $bpvf$ values, the performance of the proposed method is comparable with the FAMC and "Bici & Akar" algorithms upto $bpvf$ value of 1.5. At higher $bpvf$ values, the RD performance is better than the other three methods. Thus, the proposed SVD-based spatio-temporally scalable compression method performs better than the existing three algorithms.

3.6 Conclusions

In this chapter, we presented a novel scheme for the scalable compression of isomorphic 3D animation geometry. An encoder and a decoder structure has been proposed to obtain spatio-temporal scalability by using the SVD on geometry data. The proposed compression method applies the SVD on the animation geometry matrix $\mathbf{B}$ and selects a reduced number of spatial and temporal singular vectors for the scalable reconstruction of animation data. The number of spatial and temporal singular vectors are selected based on the input threshold value, λ_{Th}, corresponding to the cumulative signal energy contained in the singular values. The set of elements of the spatial and temporal singular vectors are further decomposed into different spatio-temporal layers to support scalability. The spatial layers are formed using the patch based mesh simplification technique. The temporal layers are formed by arranging the elements of the temporal singular vectors in a hierarchical manner by sampling them in power of 2. The prediction of elements in all the upper spatio-temporal layers is performed using averaging of neighbourhood elements in the lower layers. An encoder and a decoder structure are also presented for the proposed compression scheme. The performance of the proposed method is evaluated on four standard 3D animation sequences. The rate-distortion performance has been evaluated using the KG_{err} and the $STED$ metrics at different target bitrate values T_{bpvf}. The performance has been also compared with three existing scalable compression methods. It is observed that the proposed compression algorithm outperforms these methods in terms of the scalable rates and distortions.

Indian Institute of Technology Guwahati

4

Compression of Block-isomorphic Multi-object Animation Geometry

Contents

4.1 Introduction

The existing methods for animation geometry compression consider the entire animation as a single object. This chapter explores the geometry compression methods for multi-object based 3D (MO-3D) animations. An MO-3D animation comprises of multiple 3D objects and the mesh attributes of the constituent 3D objects change across the frames. These data require large disk space for storage and high bandwidth for real-time transmission over the network compared to single-object based 3D (SO-3D) animations. This global consideration of the multiple 3D objects as a single object does not exploit fully the local spatio-temporal redundancies exhibited by the individual 3D objects for the animation duration. Moreover, in a real-world MO-3D animation, the 3D objects may appear or disappear at irregular intervals during the animation period. In that case, the frame-wise 3D objects of the MO-3D animation will follow graph isomorphism property in blocks of consecutive frames depending on the changing mesh attributes of the 3D objects.

This chapter proposes an efficient geometry compression method for MO-3D animations by first applying a temporal segmentation algorithm on the animation data to group the consecutive frames as set of isomorphic multi-object (IMO) blocks or segments. A connectivity based spatial segmentation algorithm is then applied to each IMO block to identify the vertices belonging to the constituent 3D objects. The geometry data of the individual 3D object are then separated out across the frames within each IMO block. For MO-3D animations with a single IMO block, the principal component analysis (PCA) is applied on the geometry data of each 3D object. This object based PCA (OPCA) exploits the spatio-temporal redundancies of the individual 3D objects across the frames in a better way compared to applying the PCA globally on geometry data of all the objects together. For MO-3D animations with multiple IMO blocks, 3D objects spanning across multiple IMO blocks are detected as unique 3D objects based on the mesh attributes and the Euclidean distances. The geometry data of the unique 3D objects are appended temporally and are finally compressed using the proposed OPCA algorithm. To compare the performance of the proposed method, a few MO-3D animations are created using the existing public domain SO-3D animations. The simulation results show that the proposed object-based PCA (OPCA) method gives better performance in terms of objective and subjective quality metrics compared to the other PCA based geometry compression methods applied globally on the MO-3D animations.

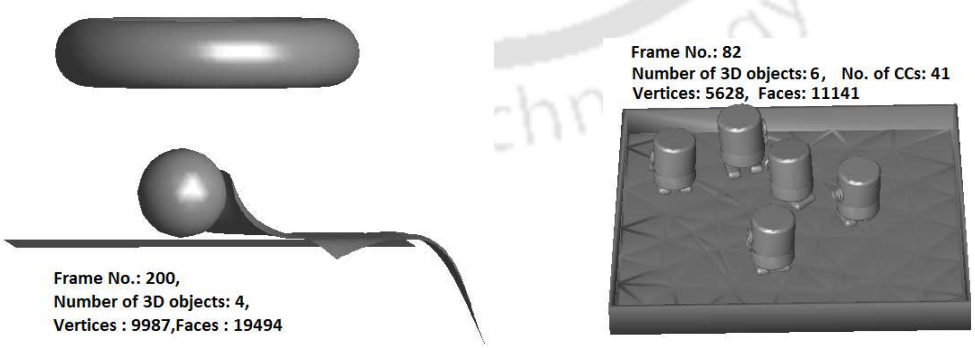
The rest of the chapter is organized as follows. Section 4.2 presents an overview of the different types of MO-3D animations along with a short review of the existing compression methods that can be applied for the geometry compression of MO-3D animations. Section 4.3 presents the motivations for the work. Section 4.4 describes the mathematical representation for MO-3D animations. Section 4.5 presents the details about the proposed object-based PCA (OPCA) geometry compression algorithm. In Section 4.6, we present the experimental results and performance of the proposed method in comparison to other compression methods. Section 4.7 draws conclusions with a discussion on the proposed work.

4.2 Classification of MO-3D animations

The MO-3D animations can be categorised based on the nature of changing mesh attributes of the constituent 3D objects across the frames as follows:

- **Isomorphic multi-object 3D animation**

  In an *isomorphic multi-object 3D* (IMO-3D) animation, the topology information of the constituent 3D objects is constant across the whole animation duration with a fixed number of vertices per frame. An IMO-3D animation comprises of identical number of 3D objects across the frames with only varying geometry data. Figure 4.1 show selected frames each from the "Cloth" and "Toasters" IMO-3D animations. Each frame of the above two animations are composed of 4 and 5 3D objects per frame respectively from start to end.



(a) A frame from "Cloth" sequence with 4 objects

(b) A frame from "Toasters" sequence with 5 objects

Figure 4.1: Examples of IMO-3D animations.

- **Block-isomorphic multi-object 3D animation**

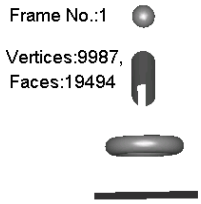
  In a *block-isomorphic multi-object 3D* (BIMO-3D) animation, the topology information of the constituent 3D objects satisfies the graph isomorphism property for a set of successive frames considered as an IMO block or segment. For each IMO block, the total number of vertices from the multiple 3D objects per frame is constant with identical connectivity data and varying geometry data across the frames within it. Whenever there are changes in the nature or number of 3D objects present in a frame during the animation length, the values of the primary mesh attributes get changed to another set of values for the set of frames belonging to another IMO block. Figure 4.2 shows the frames from a BIMO-3D animation named "Cloth-Cow-Dance" with five IMO segments. The sequence has been created by combining 3D objects from three 3D animations namely, "Cloth", "Cow" and "Dance". The first IMO segment comprises of four 3D objects and ranges from 1^{st} frame to 50^{th} frame as shown in Figures 4.2(a) and 4.2(b). Likewise, the 2^{nd}, 3^{rd}, 4^{th} and 5^{th} IMO segments are shown in Figures 4.2(c) to 4.2(d), 4.2(e) to 4.2(f), 4.2(g) to 4.2(h) and 4.2(i) to 4.2(j) respectively.

- **Non-isomorphic multi-object 3D animation**

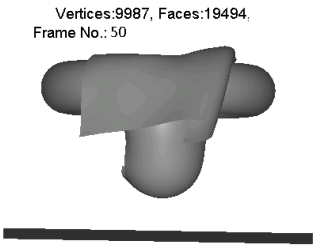
  In a *non-isomorphic multi-object 3D* (NIMO-3D) animation, the mesh attributes of the 3D objects do not follow the graph isomorphism property across the frames. In this case, the number of vertices and triangular faces of the constituted multiple 3D objects change in each frame across the animation length. They are constructed by 3D scanning of instantaneous poses of real world 3D objects over a certain duration. The frames from a NIMO-3D animation named "Dog-Man-Ball" are shown in Figure 4.3. The compression of the NIMO-3D animation geometry is complex and not considered here.

4.3 Motivation and the objectives

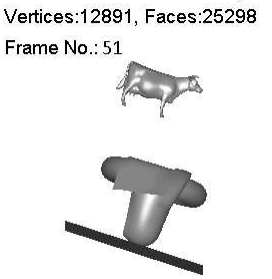
The existing 3D animation geometry compression algorithms deal only with the SO-3D animations where the mesh attributes of a single 3D object are assumed to be isomorphic throughout all the frames of the animation. In a real-world scenario, the 3D animations may contain multiple 3D objects in each frame. The geometry as well as the connectivity data of the multiple 3D objects per frame may be changing over time. The existing literature does not address the compression issues of MO-3D



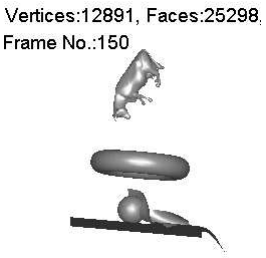
(a) 1st IMO segment with 4 objects (start)



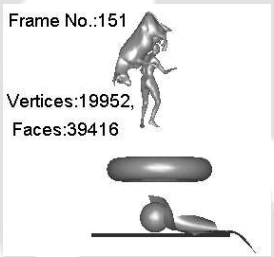
(b) 1st segment with 4 objects (end)



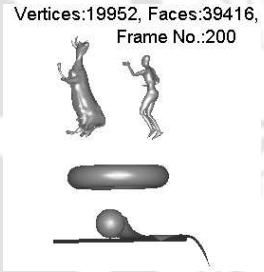
(c) 2nd IMO segment with 5 objects (start)



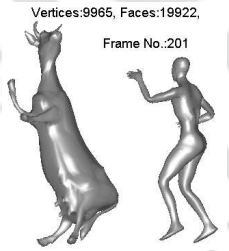
(d) 2nd IMO segment with 5 objects (end)



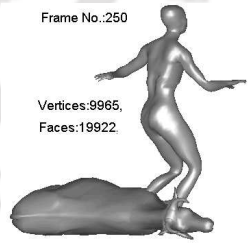
(e) 3rd IMO segment with 6 objects (start)



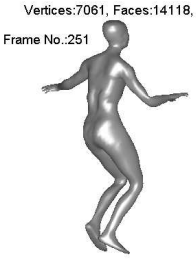
(f) 3rd IMO segment with 6 objects (end)



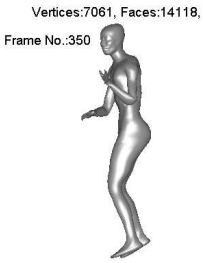
(g) 4th IMO segment with 2 objects (start)



(h) 4th IMO segment with 2 objects (end)



(i) 5th IMO segment with 1 object (start)



(j) 5th IMO segment with 1 object (end)

Figure 4.2: Example of a BIMO-3D animation ("Cloth-Cow-Dance") with 5 IMO segments

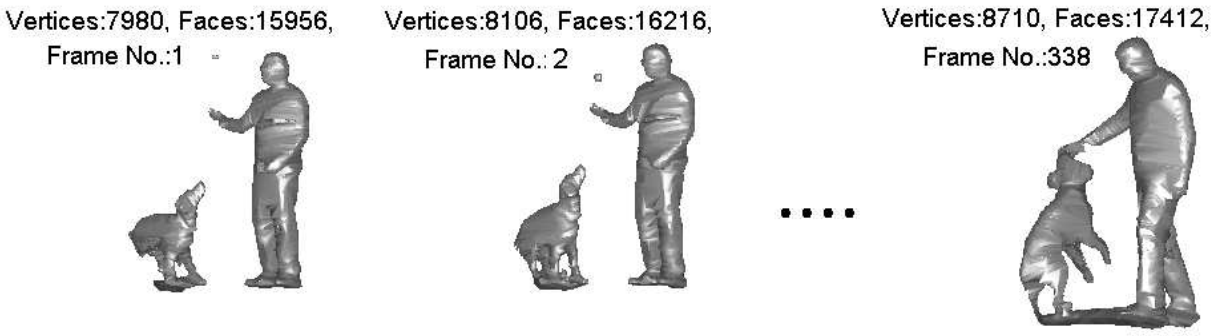


Figure 4.3: Selective frames of a NIMO-3D animation ("Dog-Ball-Man")

animations. In this work, we address the compression issues for the raw geometry data of IMO-3D and BIMO-3D animations for efficient storage and transmission over the network. The motivation and objectives for this work are based on the following facts:

(i) The 3D animation geometry may comprise a single or multiple 3D objects per frame. A single object may further have a single (or multiple) connected component(s) (CC(s)). The CCs signify the number of isolated groups of interconnected vertices in the 3D space, which are clubbed together to form a single 3D model. For example, in the "Chicken" sequence shown in Figure 4.4, the single 3D object has been created with 41 isolated CCs with varying geometry across the frames.

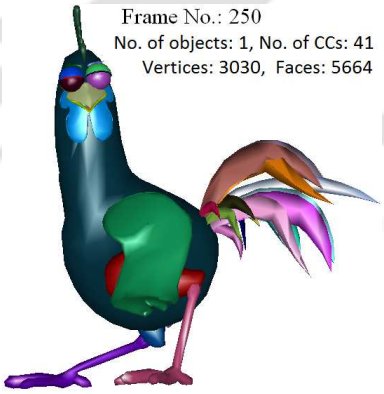


Figure 4.4: A frame from "Chicken" SO-3D animation with 41 CCs

Most of the animation geometry compression works reported in the literature deal with 3D animations featuring single 3D object having single or multiple CCs. Since the multiple CCs are

part of a single 3D object, the temporal motions of vertices belonging to each CC will be almost similar to that of the vertices in another CCs and in synchronous with the overall temporal motion of the single 3D object. Therefore, the geometry data of "Chicken" sequence can be compressed by taking the time varying geometry data from all the CCs together. However, a 3D animation may contain multiple 3D objects per frame with multiple CCs. As for example, the "Cloth" and the "Toaster" IMO-3D animations shown in Figures 4.1(a) and 4.1(b) respectively are composed of 4 and 5 different 3D objects per frame for the entire animation length. For such animations, the temporal motions of the vertices belonging to each 3D object will vary from object to object. So, the application of a geometry compression algorithm available for SO-3D animations on such IMO-3D animations by considering the geometry data of all the multiple 3D objects together is not the best choice. This approach will not exploit the redundancies present in the individual 3D object's spatial and temporal positional values across the frames. Moreover, the raw data size of such IMO-3D animations will be much higher compared to the SO-3D animation. So, the application of geometry compression algorithms available for SO-3D animations on the overall 4D geometry data (3D+time) of multiple 3D objects requires a large memory space and processing time. Thus, there is a need to address the compression issues for such IMO-3D animations.

(ii) The existing geometry compression methods reported in the literature deal only with isomorphic 3D animations. However, there may exist multiple 3D objects in each frame of 3D animations and the geometry as well as the connectivity of vertices of constituted multiple 3D objects may be changing at irregular intervals over time span. This usually happens when some of the existing 3D objects leave or new 3D objects enter some frames during the animation length. Such variations in the number of 3D objects and their mesh attributes occur in BIMO-3D animations as discussed above. The existing literature does not address the compression issues related to BIMO-3D animations. Moreover, the existing geometry compression methods available for isomorphic 3D animations can not be applied directly on the BIMO-3D animations because of changes in total number of vertices of the 3D objects and their connectivity data per frame in the sequence. So, there is a requirement to develop efficient geometry compression schemes for such BIMO-3D animations.

4.4 Representation of the MO-3D animation geometry

The mathematical representations of IMO-3D and BIMO-3D animation geometry differ based on the nature of their changing mesh attributes. They are described below.

(i) **Representation of the IMO-3D animation geometry**

Consider an IMO-3D animation with constant topology information $\mathbf{C}_d$ and dynamic geometry component $\mathbf{G}_d$ spanning over a time duration of N_f number of frames. The component $\mathbf{G}_d$ is defined over the set of N_f frames containing the geometry positions of N_v vertices describing the 3D objects in each frame. In case of IMO-3D animations, the total number of vertices from all the 3D objects per frame remain same throughout the mesh sequences. So, the representation of the animation geometry data and connectivity data for an IMO-3D animation is similar to that of an ISO-3D animation as given by Equations (1.3) and (1.1) respectively in Chapter 1. The overall geometry component $\mathbf{G}_d$ of the IMO-3D animation is also represented in a 2D matrix form for the easy application of signal processing algorithms on it. The matrix representation can follow either the frame-based or trajectory-based representation as given by Equations (1.6) and (1.8) respectively in Chapter 1.

To represent the geometry of an IMO-3D animation using the frame-based matrix format, the x, y and z coordinate positions of N_v vertices of the f-th frame are cascaded together to form a column vector $\mathbf{a_f}$ of dimension $(3N_v \times 1)$ as given by Equation (1.5). The column vectors $\mathbf{a_f}, f = 1, \ldots, N_f$ for all the N_f frames are then placed as the columns of a matrix $\mathbf{A}_{IMO}$ of dimension $(3N_v \times N_f)$ as

$$\mathbf{A}_{IMO} = \left[\mathbf{a}_1 \mid \mathbf{a}_2 \mid \ldots \mid \mathbf{a}_{N_f} \right] \tag{4.1}$$

Similarly, to represent the geometry of an IMO-3D animation using the trajectory-based matrix format, the frame-wise geometrical positions of each vertex $\mathbf{p}_{v_i}^f \in \mathbb{R}^3$ across N_f frames are cascaded together to form a trajectory vector $\mathbf{t_i}$ of dimension $(3N_f \times 1)$ as given by Equation (1.7).The trajectory vectors $\mathbf{t_i}, i = 1, \ldots, N_v$ for all the N_v vertices are then placed as the columns of a matrix $\mathbf{B}_{IMO}$ of dimension $(3N_f \times N_v)$ as

$$\mathbf{B}_{IMO} = \left[\mathbf{t}_1 \mid \mathbf{t}_2 \mid \ldots \mid \mathbf{t}_{N_v} \right] \tag{4.2}$$

(ii) **Representation of the BIMO-3D animation geometry**

In case of BIMO-3D animation, the overall animation geometry data for all the frames can not be stored directly in a single 2D matrix because of the varying number of vertices in each of the IMO block. They can be best represented by a set of matrices with each matrix representing the animation geometry data of an IMO segment defined for a group of successive frames. Using the frame-based matrix representation of the 3D animation geometry as given by Equation (1.6) in Chapter 1, the animation geometry $\mathbf{A}_{BIMO}$ for a BIMO-3D animation is written as

$$\mathbf{A}_{BIMO} = \{\mathbf{A}_1, \ldots, \mathbf{A}_i, \ldots, \mathbf{A}_{N_b}\} \tag{4.3}$$

where $\mathbf{A}_i \in \mathbb{R}^{3N_{v_i} \times N_{f_i}}$ is the frame-based animation geometry matrix for the i-th IMO segment with length of N_{f_i} frames and N_b is the number of IMO blocks in the BIMO-3D animation. Similarly, using the trajectory-based matrix representation of the 3D animation geometry as given by Equation (1.8) in Chapter 1, the geometry of a BIMO-3D animation can be also represented by a set of matrices and is written as

$$\mathbf{B}_{BIMO} = \{\mathbf{B}_1, \ldots, \mathbf{B}_i, \ldots, \mathbf{B}_{N_b}\} \tag{4.4}$$

where $\mathbf{B}_i \in \mathbb{R}^{N_{v_i} \times 3N_{f_i}}$ is the trajectory-based animation geometry matrix for the i-th IMO block with length of N_{f_i} frames. The connectivity data for the BIMO-3D animation are represented by a set of connectivity matrices representing the IMO blocks and written as

$$\mathbf{C}_{BIMO} = \left\{\mathbf{C}_{d_1}, \ldots, \mathbf{C}_{d_2}, \ldots, \mathbf{C}_{d_{N_b}}\right\} \tag{4.5}$$

where $\mathbf{C}_{d_i} \in \mathbb{R}^{N_{t_i} \times 3}$ is the connectivity matrix for i-th IMO block as defined using Equation (1.1).

4.5 Proposed geometry compression method of BIMO-3D animations

We propose a geometry compression scheme for the BIMO-3D animations featuring multiple 3D objects per frame. A BIMO-3D animation contains a number of IMO blocks or segments, satisfying the graph-isomorphism property as discussed earlier. To compress the geometry data of a BIMO-

3D animation, it is proposed to segment the animation data along the frames into different IMO blocks. To achieve this, a temporal segmentation algorithm has been proposed to divide a BIMO-3D animation into a set of IMO blocks based on the mesh attributes of 3D objects in each frame. After the temporal segmentation, the individual 3D objects in each IMO segment are detected using a connectivity based spatial segmentation algorithm. This has been done to exploit the redundancies present in individual 3D object's spatial and temporal positional values across the frames. Following the spatial segmentation process, the geometry data of all the detected 3D objects are extracted for each IMO segment. The geometry data of the unique 3D objects, which are present across multiple IMO blocks are merged temporally based on the number of vertices, the number of faces and the Euclidean distances. If a unique 3D object has number of vertices smaller than a threshold value for minimum vertex count per object, its geometry data across the frames are spatially merged with that of other object based on the minimum Euclidean distance. Finally, the geometry data all the unique 3D objects across the frames of all IMO blocks are compressed by applying the PCA. We term this method as *object based PCA* (OPCA). In the OPCA algorithm, the number of eigen vectors (EVs) for each object is selected based on an input threshold value of cumulative energy in the eigen values. The object-wise transformed coefficients (TCs) are obtained by projecting the object's geometry on the space spanned by the selected EVs. Finally, elements of all the EVs, TCs and the mean trajectory (MT) vector from the OPCA algorithm are uniformly quantized and entropy coded to get the compressed bitstream.

4.5.1 OPCA encoder

The block diagram of the encoder structure of the proposed OPCA based geometry compression scheme is shown in Figure 4.5. The functioning of the different blocks in the encoder is presented below:

- ***Temporal segmentation of frames into IMO segments***

  The presence of multiple 3D objects with varying mesh attributes at random intervals across the frames, makes it difficult to store the whole animation geometry data of a BIMO-3D animation in single 2D matrix as applicable for an IMO-3D animation. Thus, it is not possible to apply any PCA or CPCA based method globally on the geometry data. To overcome this, a temporal

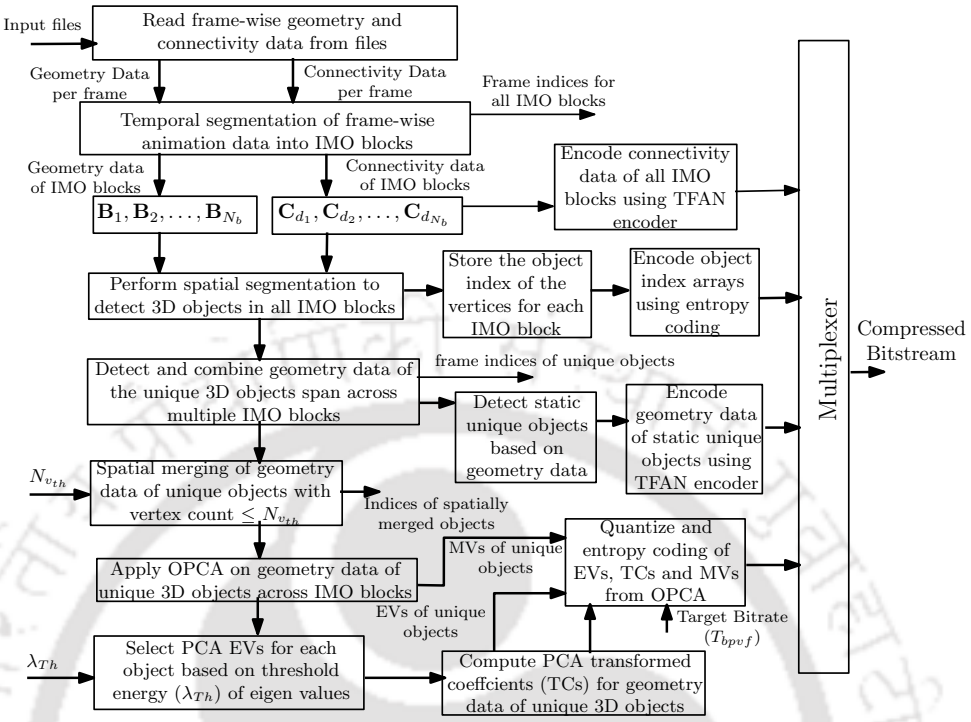


Figure 4.5: Encoder structure of proposed compression method for BIMO-3D animation

segmentation algorithm is proposed. This algorithm segments the consecutive frames into IMO blocks. The pseudocode of the proposed temporal segmentation technique is shown in Algorithm 4.1. The algorithm reads the total count of mesh attributes like number of vertices, $N_v(i)$, and number of faces, $N_t(i)$, from N_f files representing the frames of a BIMO-3D animation. The animation data of consecutive frames are considered as a part of the same IMO block until there is no change in product of total vertex count and face count per frame, $N_v(i) \times N_t(i)$. Whenever, there is a change in either of total number of vertices or faces in a frame, it is marked as the start of a new IMO block. The previous frame is marked as the end of last IMO block. The geometry and connectivity data for i-th IMO block are stored separately as $\mathbf{B}_i$ and $\mathbf{C}_{d_i}$ respectively. The frame indices of detected IMO blocks are sent as the header information to the decoder.

- ***Spatial segmentation of objects in IMO blocks***

  As shown in the block diagram, the individual 3D objects in the IMO blocks are detected using a spatial segmentation algorithm. We propose an object segmentation algorithm based on the connectivity data. The algorithm detects the vertices which are connected as a single connected

Algorithm 4.1: Detecting IMO blocks from BIMO-3D animations

Input : A sequence of N_f files for each frame of a BIMO-3D sequence, $file_1$, $file_2$, ..., $file_{N_f}$

`/* Initialization */`

$i \leftarrow 1$ `// file counter`
$n \leftarrow 1$ `// IMO block counter`
$flag\_imo \leftarrow 0$ `// IMO block detection flag`

`/* Begin with reading the 1st frame data as the reference */`
$imo\_frame\_start\_loc(n) \leftarrow i$ `// Store start frame number for an IMO block`
Read geometry data $\mathbf{G}_i$ and connectivity data $\mathbf{T}_i$ from $file_i$
Compute vertex count $N_v(i)$ from $\mathbf{G}_i$ and triangular faces count $N_t(i)$ from $\mathbf{T}_i$
Append $\mathbf{G}_i$ to matrix $\mathbf{B}_n$ as columns
$\mathbf{C}_{d_n} \leftarrow \mathbf{T}_i$ `// store connectivity data for IMO block`
$i \leftarrow i + 1$
while $i \leq N_f$ **do**
    Read geometry data $\mathbf{G}_i$ and connectivity data $\mathbf{T}_i$ from $file_i$
    Compute vertex count $N_v(i)$ from $\mathbf{G}_i$ and triangular faces count $N_t(i)$ from $\mathbf{T}_i$
    if $(N_t(i) \times N_v(i) \neq N_t(i-1) \times N_v(i-1))$ **then**
        $flag\_imo \leftarrow 1$
    else
        $\mathbf{D}_t \leftarrow \mathbf{T}_i - \mathbf{T}_{i-1}$
        Compute ΔT as Sum of elements of $\mathbf{D}_t$
        if $\Delta T \neq 0$ **then**
            $flag\_imo \leftarrow 1$
        end
    end
    if $(flag\_imo = 1)$ **then**
        $n \leftarrow n + 1$
        $imo\_frame\_start\_loc(n) \leftarrow i$
        $\mathbf{C}_{d_n} \leftarrow \mathbf{T}_i$
        $flag\_imo \leftarrow 0$
    end
    Append $\mathbf{G}_i$ to matrix $\mathbf{B}_n$ as columns
    $i \leftarrow i + 1$
end

Output : Two sets of matrices $\mathbf{B}_1, \ldots, \mathbf{B}_n$ and $\mathbf{C}_{d_1}, \ldots, \mathbf{C}_{d_n}$ representing geometry and connectivity data respectively for n IMO blocks, a set of frame indices for n IMO blocks

component (CC). A single connected component may be a whole 3D object or it may be a part of a 3D object with multiple CCs. The algorithm starts with finding the level-1 neighbouring vertices of all the vertices in i-th IMO block. In the next step, an iterative procedure is applied to

Algorithm 4.2: Detection of 3D objects in an IMO segment

Input : The connectivity matrix $\mathbf{C}_{d_i}$ and the vertex count N_{v_i} corresponding to i-th IMO segment

```
/* Initialization                                                    */
```

Set $j \leftarrow 0$ // count for each detected 3D object or connected component (CC)

Set $CC_i^j \leftarrow \{\emptyset\}$ // Sets for vertex indices of individual 3D object or CC

Set $CV_{list} \leftarrow \{\emptyset\}$ // Set for all vertex indices of the detected 3D objects or CCs

$V_{left} \leftarrow \{1, \ldots, N_{v_i}\}$ // set of all vertex indices not included in any 3D object

Set $V_{visit}(k) \leftarrow 0 \ \ \forall k \in \{1, \ldots, N_{v_i}\}$ // Set flag for all vertices as non-visited

Set $OI_i(k) \leftarrow 0 \ \ \forall k \in \{1, \ldots, N_{v_i}\}$ // Set object index for all vertices as 0

```
/* Begin                                                             */
```

Compute the set of neighbours $Vn\_L1(k)$ for each vertex $k \in \{1, \ldots, N_{v_i}\}$ using $\mathbf{C}_{d_i}$

$Vn\_new = Vn\_L1$

while $|V_{left}| \neq 0$ **do**
  $V_{left} \leftarrow V_{left} \setminus CV_{list}$
  for $k \in V_{left}$ **do**
    if $V_{visit}(k) = 0$ **then**
      $Neigh\_V_k \leftarrow Vn\_new(k)$
      $tmp \leftarrow Neigh\_V_k$
      for $v \in Neigh\_V_k$ **do**
        $V_{visit}(v) \leftarrow 1$
        $tmp \leftarrow tmp \cup Vn\_L1(v)$
      end
      $Vn\_L1(k) \leftarrow tmp \setminus \{k\}$
      $Vn\_new(k) \leftarrow Vn\_L1(k) \setminus Neigh\_V_k$
      if $(|Vn\_new(k)| = 0) \,\&\, (k \notin CV_{list})$ **then**
        $j \leftarrow j + 1$
        $CC_i^j \leftarrow Neigh\_V_k \cup \{k\}$
        $CV_{list} \leftarrow CV_{list} \cup CC_i^j$
      end
      $V_{visit}(k) \leftarrow 1$
    end
  end
end
for $l \leftarrow 1$ ***to*** j **do**
  for $n \in CC_i^l$ **do**
    $OI_i(n) \leftarrow l$
  end
end
return OI_i

Output : An array OI_i containing object indices for the N_{v_i} vertices in i^{th} IMO block

group the level-1 neighborhood vertices for the vertices forming an edge and marked the grouped vertices with a visited flag. This iterative grouping of vertices based on connected edges goes on till all the vertices connected by a single CC are visited to detect a 3D object. The procedure is repeated for the remaining vertices not belonging to any of the already detected objects. The algorithm ends when all the vertices in the IMO block are assigned to the corresponding 3D objects. The pseudo-code of the proposed spatial segmentation algorithm is presented in Algorithm 4.2. The algorithm outputs an array for object indices, OI_i, of size equal to the number of vertices per frame for i-th IMO block. Each element of OI_i array stores the object index number corresponding to a vertex in the i-th IMO block. The geometry data of the detected 3D objects is extracted across the frames in an IMO block based on OI_i. The connectivity data of individual 3D objects in each IMO block are also generated from the connectivity data $\mathbf{C}_{d_i}$.

- ***Detection and merging of 3D objects across IMO blocks***

  The spatial segmentation gives geometry and connectivity data of individual 3D objects for the frames in each IMO block. However, there may be geometry data of 3D objects which are present across multiple IMO blocks. These unique objects which exist in multiple IMO blocks are identified based on the number of vertices, the number of faces and the Euclidean distances between the objects in the last frame of the i-th IMO block and the first frame of the $i+1$-th IMO block. The geometry data of these unique 3D objects are merged temporally across the IMO blocks.

- ***Detection and encoding of static 3D objects***

  There may exists some unique 3D objects having constant geometry data across all the frames in multiple IMO blocks. These static unique 3D objects are detected using their motion characteristics and are compressed as static 3D mesh by encoding the geometry and connectivity data using the TFAN encoder described in Section 1.3.6.

- ***Spatial merging of unique 3D objects***

  For the unique 3D objects which have number of vertices $\leq N_{v_{th}}$, an input threshold for vertex count per 3D object, the geometry data of those unique 3D objects are merged with the nearest unique object based on their Euclidean distances. The frame indices for unique 3D objects

present in multiple IMO blocks and the object indices for spatially merged unique objects are sent as header information to the decoder.

- ***Object-based PCA (OPCA)***

  The PCA is applied on geometry data of each unique 3D object for the frames in multiple IMO blocks to get the required compression. The number of eigen vectors (EVs) selected for the PCA on each unique object is based on a user input threshold value λ_{Th}. The threshold value corresponds to the percentage of cumulative energy associated with the eigen values sorted in a descending order. The PCA transformed coefficients (TCs) for each unique 3D object are then computed by projecting its geometry data on the selected EVs.

- ***Quantization and Entropy Coding***

  The elements of object-wise mean vectors (MVs), the EVs and TCs obtain from the OPCA algorithm are uniformly quantized. The rate distortion (RD) analysis [43] is applied to optimise the number of quantization bits required to uniformly quantize each TC, each component of EV and MVs for an user input target bitrate. The quantized EVs, TCs and MVs for each object are then entropy coded with an arithmetic coder [12]. The arrays, OI_is, containing the object indices for the vertices in the IMO blocks are also encoded using an arithmetic coder at the encoder. The connectivity data for all the IMO blocks are encoded using the low complexity TFAN encoder. Finally, the entropy coded EVs, TCs and MVs of each unique objects are multiplexed with the encoded connectivity data and encoded object index vectors to get the compressed bit-stream.

4.5.2 OPCA decoder

The block diagram of the decoder structure of the proposed OPCA method is shown in Figure 4.6. At the decoder, the following steps are performed:

- Separate the encoded connectivity data, OPCA data (EVs, TCs and MVs) of the unique 3D objects, compressed data of static unique objects and the object indices, OI_is, of vertices in each IMO block using a demultiplexer.

- Decode the OPCA data of unique 3D objects using arithmetic decoder. De-quantize the OPCA data to get the reconstructed EVs, TCs and MVs of each unique object.

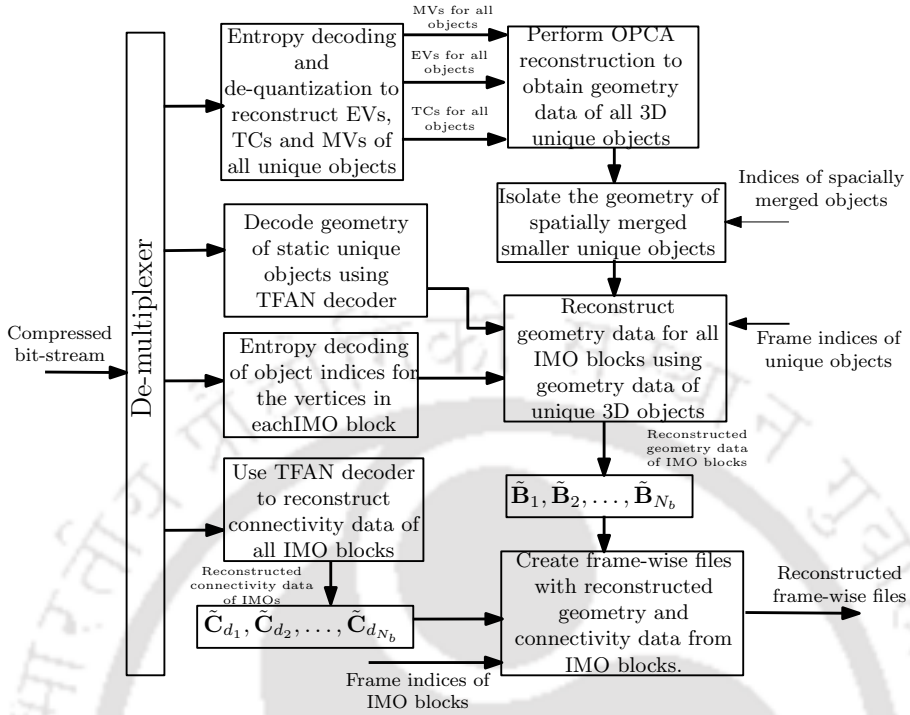


Figure 4.6: Decoder structure of proposed compression method for BIMO-3D animation

- Perform the OPCA reconstruction on individual unique objects using the de-quantized EVs, TCs and MVs to obtain the reconstructed geometry data of the each unique objects spanned across multiple IMO blocks.
- Separate the geometry data of spatially merged small unique objects using the indices of spatially merged objects.
- Decode the geometry data of static unique objects using the TFAN decoder.
- Decode the array of object indices, OI_is, representing the vertices in each IMO block using an arithmetic decoder.
- Obtain the geometry data of each IMO block using the reconstructed geometry data of unique 3D objects, reconstructed geometry of static unique objects, the frame indices of the unique objects and the array of object indices, OI_is. The reconstructed geometry data for IMO blocks are represented as $\tilde{\mathbf{B}}_1, \tilde{\mathbf{B}}_2, \ldots, \tilde{\mathbf{B}}_{N_b}$.
- Decode the connectivity data of each IMO block using the TFAN decoder and the reconstructed

connectivity data for the all the IMO blocks are represented as $\tilde{\mathbf{C}}_{d_1}, \tilde{\mathbf{C}}_{d_2}, \ldots, \tilde{\mathbf{C}}_{d_{N_b}}$.

- Generate the frame-wise reconstructed animation files for the BIMO-3D animation by combining the reconstructed geometry data and the connectivity data of IMO blocks based on frame indices of IMO blocks.

4.6 Experimental results

In this section, experimental results are presented to evaluate the performance of the geometry compression algorithms on IMO-3D and BIMO-3D animations. The compression performance has been evaluated by measuring the distortion between the original and the reconstructed animations in terms of the objective and the subjective metrics at different target bitrates measured in $bpvf$. The objective metric, KG_{err}, and the subjective metric, $STED$, have been used to quantify the distortions between the original and the reconstructed animations. Three sets of experiments are performed to compare the results of the geometry compression methods proposed for IMO-3D and BIMO-3D animations as follows:

4.6.1 Compression performance of OPCA, PCA and CPCA algorithms on the IMO-3D animations

This experiment investigates the compression performance of the proposed OPCA, the direct PCA and the CPCA methods applied on IMO-3D animations. Some of these IMO-3D animations namely, "Cloth", "Toasters" and "Marbles" are available in public domain. A few IMO-3D animations like, "Cow-Dance-Snake", "Cloth-Cow-Dance" etc., are also created using existing standard 3D animations to evaluate the performance. The properties of the five IMO-3D animations used in evaluation of the above algorithms are given in the Table 4.1.

The proposed segmentation algorithm is applied to segment the objects on the IMO-3D animations without spatial merging. The segmentation result for the "Cow-Dance-Snake" IMO-3D animation is shown in Figure 4.7. Figure 4.7(a) shows the detected 3D objects in a frame. The detected objects are shown in different colours. The extracted individual objects "Cow", "Dance", and "snake" are shown in Figures 4.7(b), 4.7(c) and 4.7(d) respectively. The segmentation results are found to be satisfactory with other IMO-3D sequences also.

To evaluate the compression performance of the OPCA method, four IMO-3D animations are used.

Table 4.1: Details of IMO-3D animation sequences

Name of MO-3D Sequence	Vertices (N_v)	Frames (N_f)	Faces (N_t)	No. of objects per frame
Cloth	9987	200	19494	4
Cow-Dance-Snake	19144	134	38276	3
Cloth-Cow-Dance	19952	200	39416	6
Toasters	5628	246	11141	41
Marbles	4620	500	8800	110

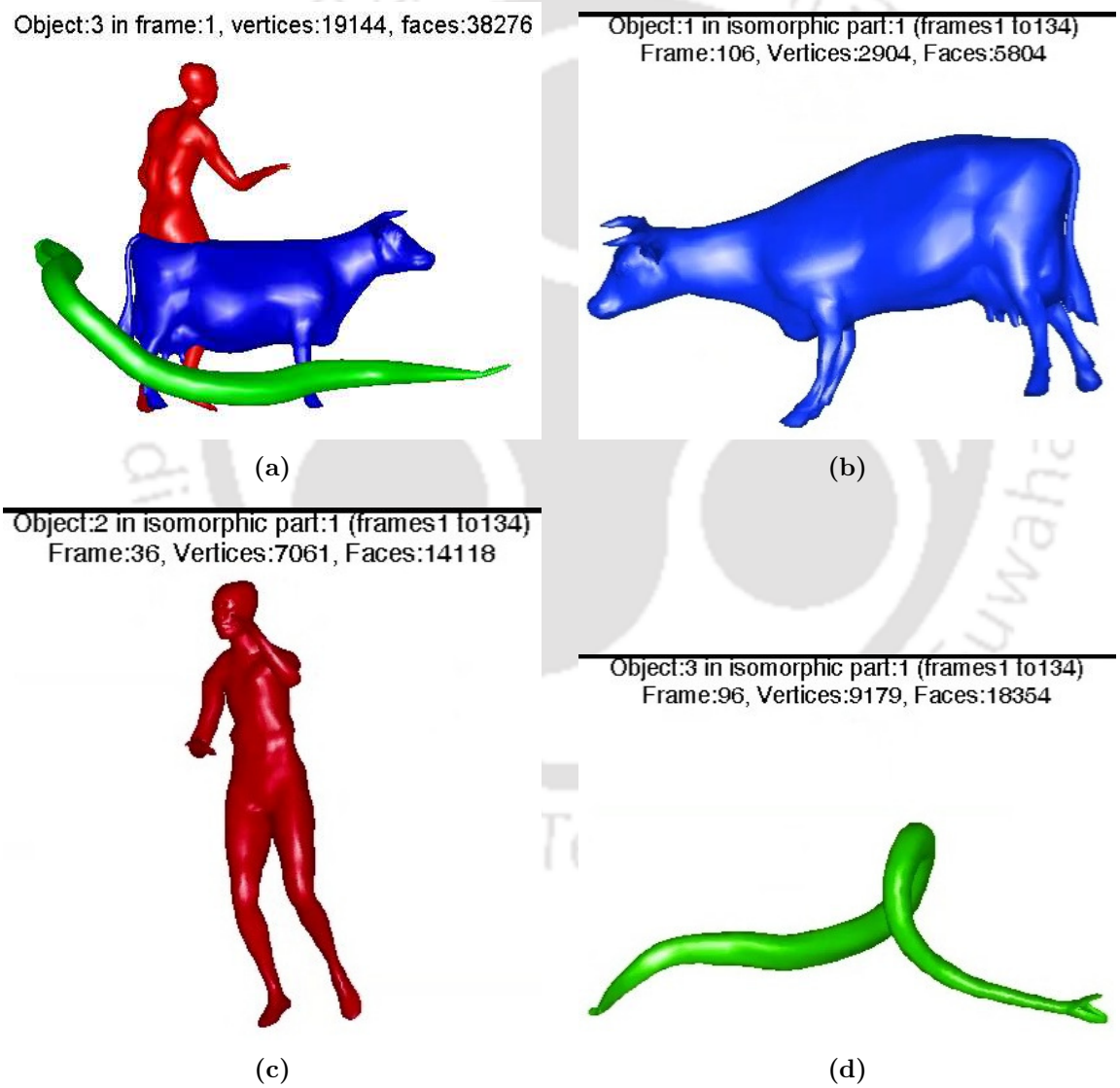


Figure 4.7: Detected and segmented objects from a frame of "Cow-Dance-Snake" IMO-3D animation sequence: (a) a frame with detected three objects, (b) segmented object "Cow", (c) segmented object "Dance" and (c) segmented object "Snake".

The number of EVs for each object has been selected for a threshold value, $\lambda_{Th} = 99.9\%$ corresponding to cumulative energy of the eigen values. The same threshold value is also used for the direct PCA and the CPCA method. For the CPCA algorithm, the number of clusters is considered same as the number of 3D objects present in the IMO-3D animations. Figures 4.8(a), 4.8(b), 4.8(c) and 4.8(d) show the performance comparison of the OPCA, the PCA and the CPCA methods. Each figure shows the plot of KG_{err} values obtained at different target bitrates ($bpvf$).

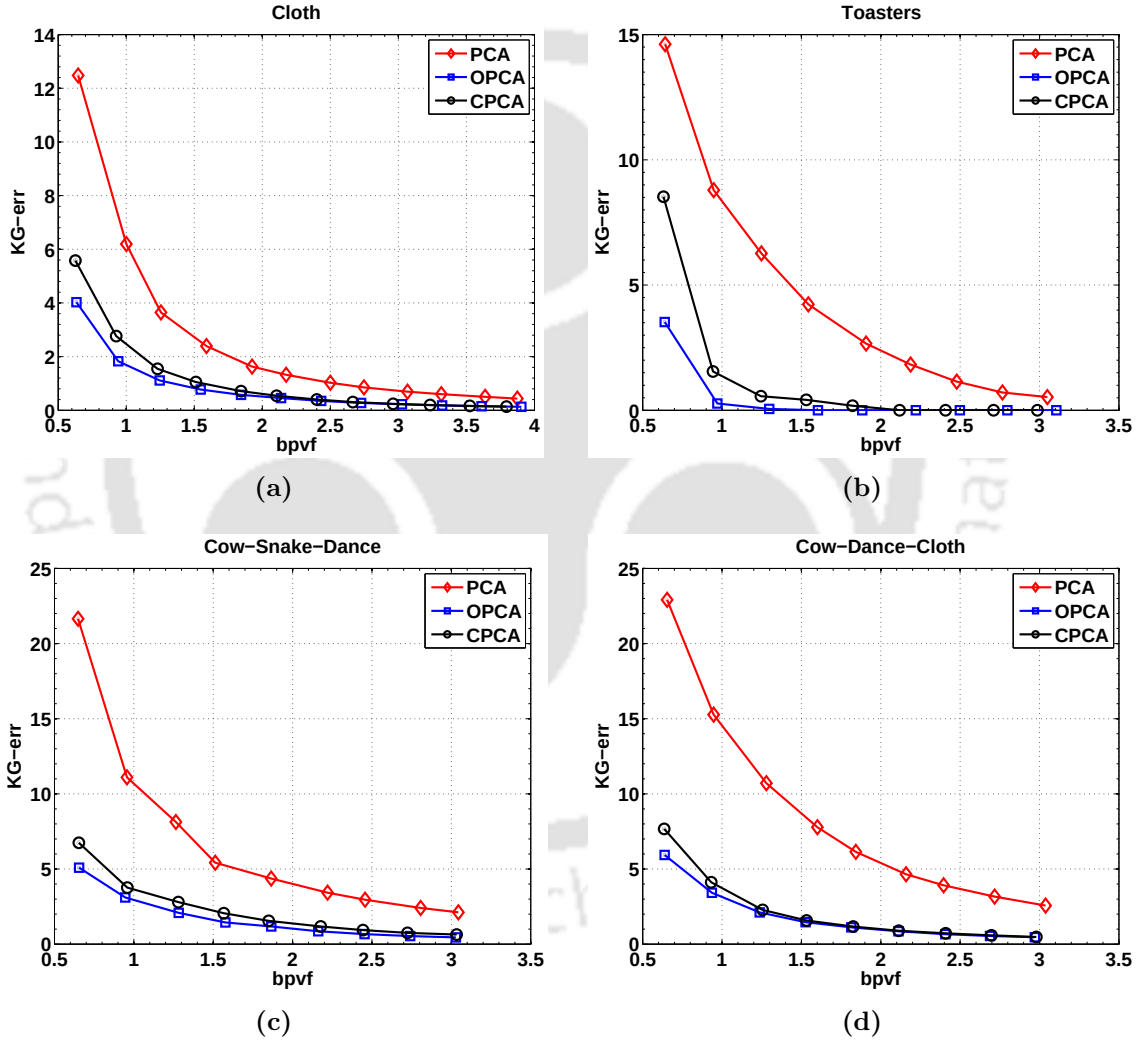


Figure 4.8: Comparative performance of the PCA, the OPCA and the CPCA algorithms in terms of the KG_{err} metric on a few IMO-3D animations: (a) "Cloth" sequence with 3 objects, (b) "Toasters" sequence with 41 objects, (c) "Cow-Dance-Snake" sequence with 3 objects and (d) "Cloth-Cow-Dance" sequence with 3 objects.

It is observed that at lower $bpvf$ values, the proposed OPCA method gives better results by

providing minimum distortion values compared to the PCA and the CPCA method. At higher $bpvf$ values, the OPCA method gives almost similar performance as the CPCA method. Similarly, Figures 4.9(a), 4.9(b), 4.9(c) and 4.9(d) show the performance comparison of the OPCA, the PCA and the CPCA methods in terms of $STED$ metric.

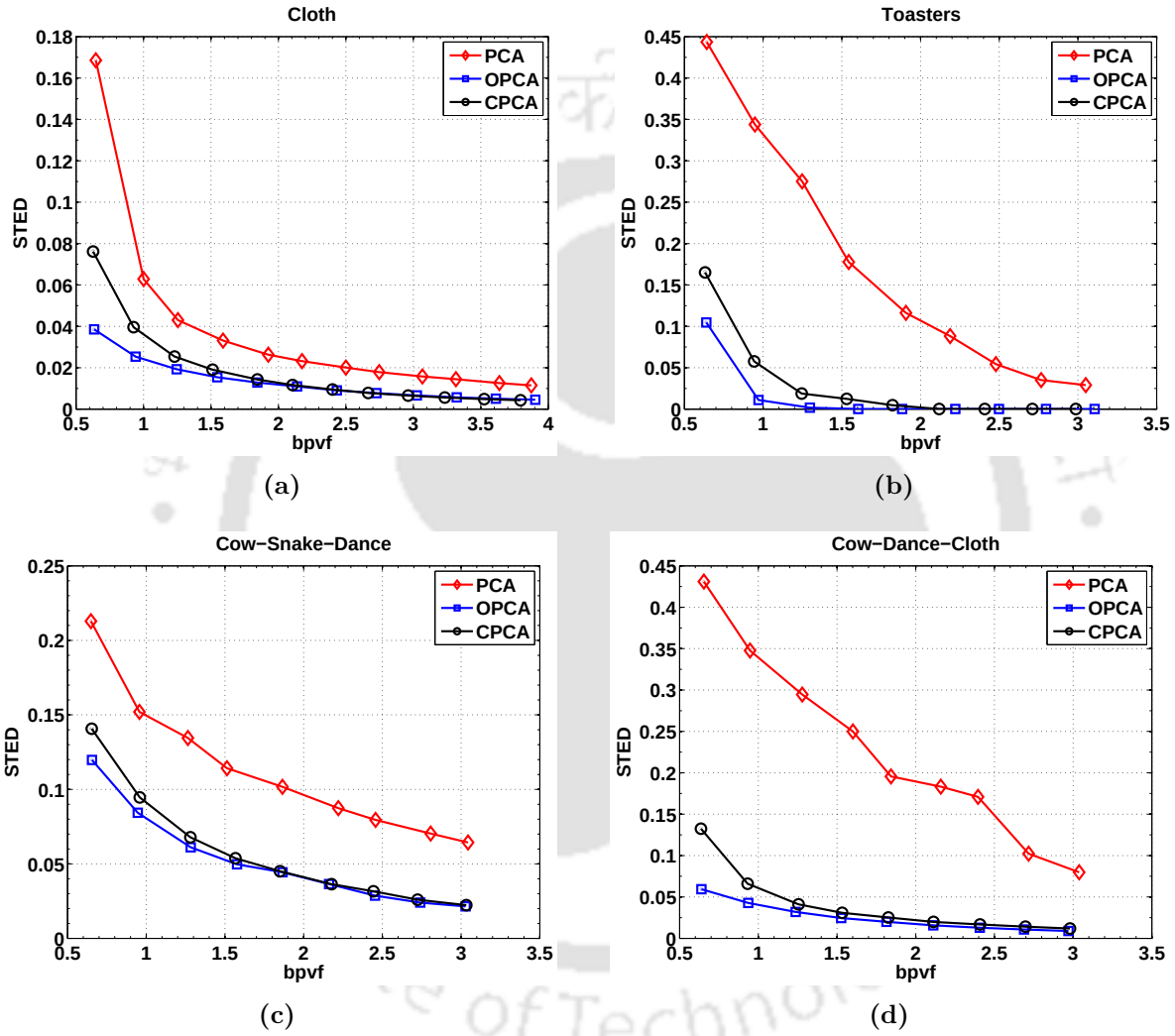


Figure 4.9: Comparative performance of the PCA, the OPCA and the CPCA algorithms in terms of $STED$ metric on a few IMO-3D animations: (a) "Cloth" sequence with 3 objects, (b) "Toasters" sequence with 41 objects, (c) "Cow-Dance-Snake" sequence with 3 objects and (d) "Cloth-Cow-Dance" sequence with 3 objects.

Each plot shows the values of $STED$ metric obtained at different target $bpvf$ values. At lower $bpvf$ values, the $STED$ metrics for the OPCA method are better compared to the PCA and the CPCA method. But, at higher $bpvf$ values, the OPCA method gives similar performance as the CPCA method. The visual performance of the OPCA method is shown in Figure 4.10 by plotting

an original frame and the reconstructed frame for the "Cow-Dance-Snake" IMO-3D animation. No perceivable visible distortions are present in the reconstructed frame.

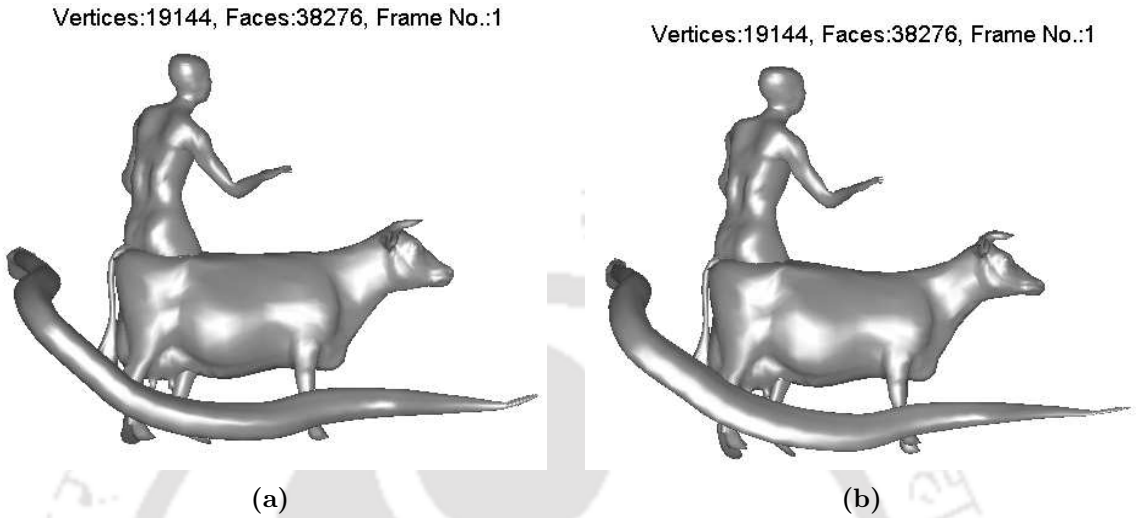


Figure 4.10: Visual performance of the OPCA method : (a) An original frame and (b) the reconstructed frame ("Cow-Dance-Snake" sequence).

4.6.2 Compression performance of SA-PCA and G-PCA for BIMO-3D animations

In this experiment, the temporal segmentation algorithm (as given in Algorithm 4.1)) is applied across the frames of a BIMO-3D animation to divide the animation data into IMO blocks. The PCA is applied on the geometry data of each IMO block. This method is termed as *segment adaptive PCA* (SA-PCA). Experimentation is also carried out to apply the PCA globally on a BIMO-3D animation after zero padding the geometry data in each frame. This is because of varying number of vertices and faces across the frames at irregular intervals and to make the vertices equal in numbers in each frame. The operation converts geometry of a BIMO-3D animation into an IMO-3D animation with a single IMO block. The PCA is applied on the single IMO block. We term this method as global PCA (G-PCA). The compression performance of above two methods are evaluated on some BIMO-3D animations. These sequences are created using the available SO-3D and IMO-3D animation databases. The properties of the BIMO-3D animations are listed in Table 4.2. To evaluate the compression performances of the SA-PCA and G-PCA, four BIMO-3D animations are used. A threshold value, $\lambda_{Th} = 99.9\%$, is used to select the EVs for both the methods. The RD performances are measured in terms of the KG_{err} and $STED$ metrics for different target bitrates ($bpvf$). Figures 4.11(a), 4.11(b)

Table 4.2: Details of BIMO-3D animation sequences

Name of BIMO-3D Sequence	Vertices segment-wise (N_v)	Frames segment-wise (N_f)	Faces segment-wise (N_t)	No. of objects segment-wise
Camel-Horse	8431,21887	48,48	16843,43814	1,1
Cloth-Toasters	9987,5628	200,246	19494,11141	4,41
Cow-Dance-Snake	2904,7061,9179	200,200,134	5804,14118,18354	1,1,1
Cloth-Cow-Dance	9987,2904,7061	200,200,200	19494,5804,14118	4,1,1

4.11(c), and 4.11(d) show the performance comparison of the SA-PCA and the G-PCA methods.

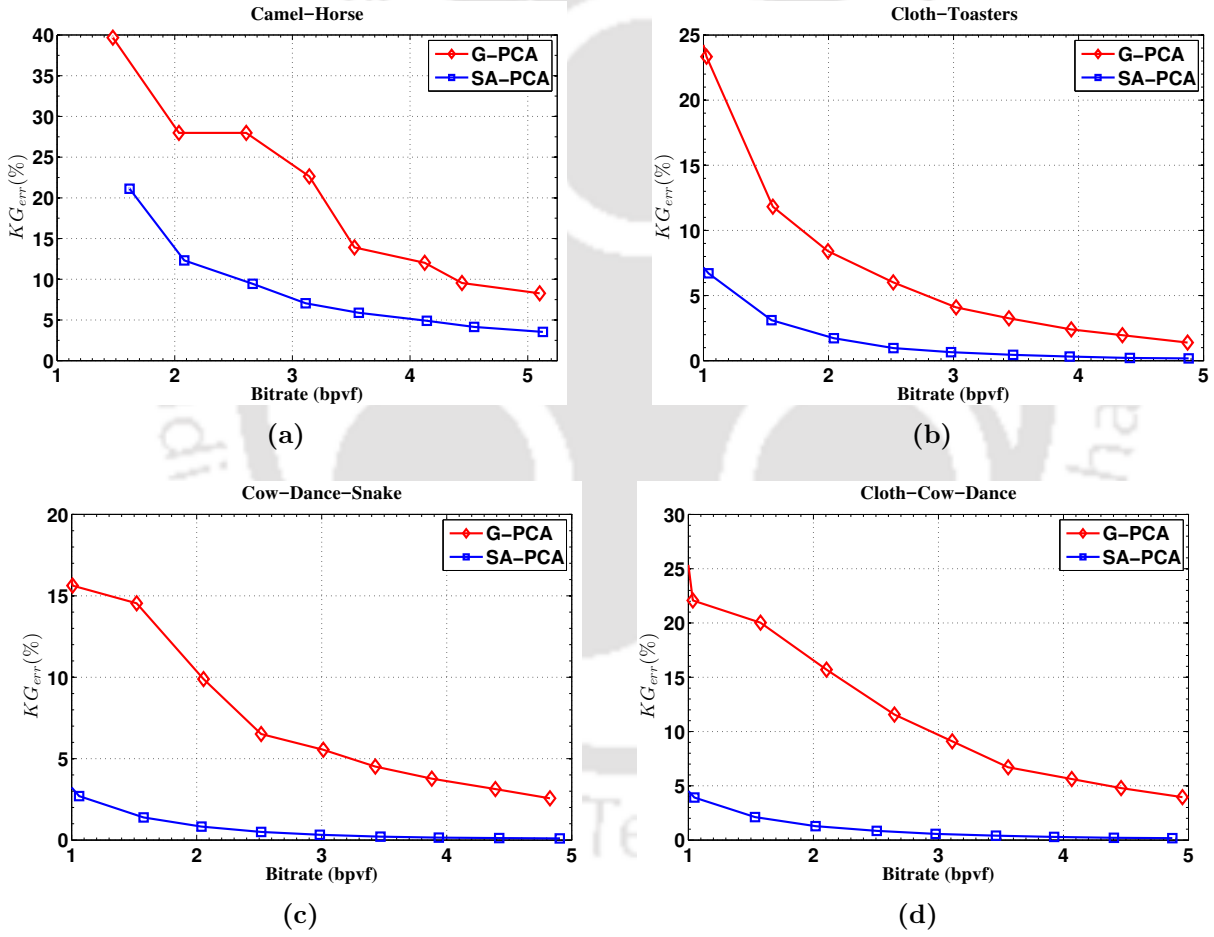


Figure 4.11: Comparative performance of the G-PCA and the SA-PCA algorithms in terms of KG_{err} metric on a few BIMO-3D animations: (a) "Camel-Horse" sequence with 2 IMO blocks, (b) "Cloth-Toasters" sequence with 2 IMO blocks, (c) "Cow-Dance-Snake" sequence with 3 IMO blocks and (d)"Cloth-Cow-Dance" sequence with 3 IMO blocks.

It is observed that at lower $bpvf$ values, the SA-PCA method gives better performances by pro-

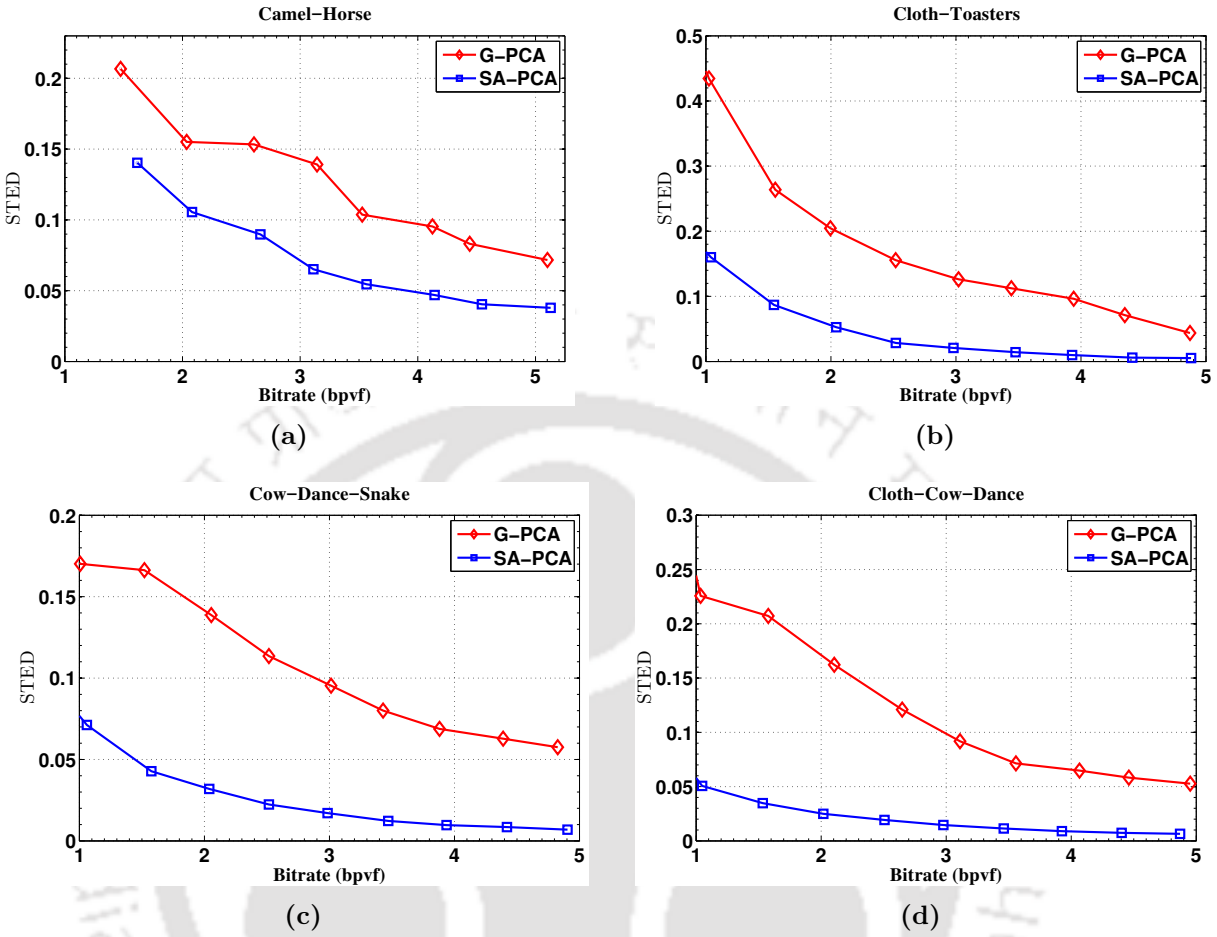


Figure 4.12: Comparative performance of the G-PCA and the SA-PCA algorithms in terms of *STED* metric on a few BIMO-3D animations: (a) "Camel-Horse" sequence with 2 IMO blocks, (b) "Cloth-Toasters" sequence with 2 IMO blocks, (c) "Cow-Dance-Snake" sequence with 3 IMO blocks and (d)"Cloth-Cow-Dance" sequence with 3 IMO blocks.

viding minimum distortion values compared to the G-PCA. Similar results are obtained for RD performance measured using the *STED* metric as shown in Figure 4.12.

4.6.3 Compression performance of SA-OPCA and OPCA for BIMO-3D animations

In this experiment, the compression performance for the BIMO-3D animations has been evaluated by applying the OPCA geometry compression algorithm on the geometry data. We have applied the OPCA algorithm in two different ways. (i) In the first case, the temporal segmentation algorithm is applied across the frames of a BIMO-3D animation to segment it into IMO blocks. The OPCA is

applied on the animation data of each IMO block. This method is termed as segment adaptive OPCA (SA-OPCA). (ii) In the second case, spatial segmentation is applied on all IMO blocks to isolate the geometry data of the constituent 3D objects. The geometry data of the unique 3D objects present in multiple IMO blocks are extracted across the frames in IMO blocks. The geometry data of the unique 3D objects which are very small in number of vertices, are further merged spatially with the other unique objects based on their Euclidean distances and a threshold value for the vertex count. The OPCA algorithm is then applied on the geometry data of all the merged unique 3D objects. This method is termed as merged OPCA (M-OPCA). The geometry compression results obtained using the M-OPCA method are compared with those of SA-PCA and SA-OPCA algorithms. Figure 4.13 shows the segmented 3D objects in a frame of each IMO block obtained from the "Cloth-Cow-Dance" BIMO-3D animation. The vertices belonging to the segmented 3D objects are plotted in separate colours for visual distinction. To evaluate the compression performances of the SA-PCA, SA-OPCA and M-OPCA methods, four BIMO-3D animations namely, "Chicken-Cloth", "CowDanceSnake", "ClothToasters" and "ClothCowDance" are considered. A threshold value, $\lambda_{Th} = 99.9\%$ is used for all the methods to select the eigen values and their corresponding EVs. The RD performances are measured in terms of KG_{err} and $STED$ metrics for different target bitrates ($bpvf$s). Figures 4.14(a), 4.14(b) 4.14(c) and 4.14(d) show the performance comparison of the SA-PCA, SA-OPCA and M-OPCA methods in terms of KG_{err} metric for different $bpvf$s. The RD-performance of the SA-PCA is inferior to other two methods for all the sequences. The best performance is found for the M-OPCA method for all the sequences at lower $bpvf$ values. At higher $bpbf$ values, the performance of SA-OPCA and M-OPCA methods are almost similar. The RD performances of the algorithms on the above four sequences are measured using the $STED$ metric as shown in Figure 4.15. The RD-performance of the SA-PCA shows inferior to other two methods for all the sequences. For the SA-OPCA and M-OPCA methods, the compression performances are similar as shown in Figures 4.15(a), 4.15(c) and 4.15(d). For the "CowDanceSnake" sequence, the performance of M-OPCA is much better than SA-OPCA and SA-PCA as shown in Figure 4.15(b). Thus, the experimental results demonstrate the superior performance of the proposed object based compression schemes.

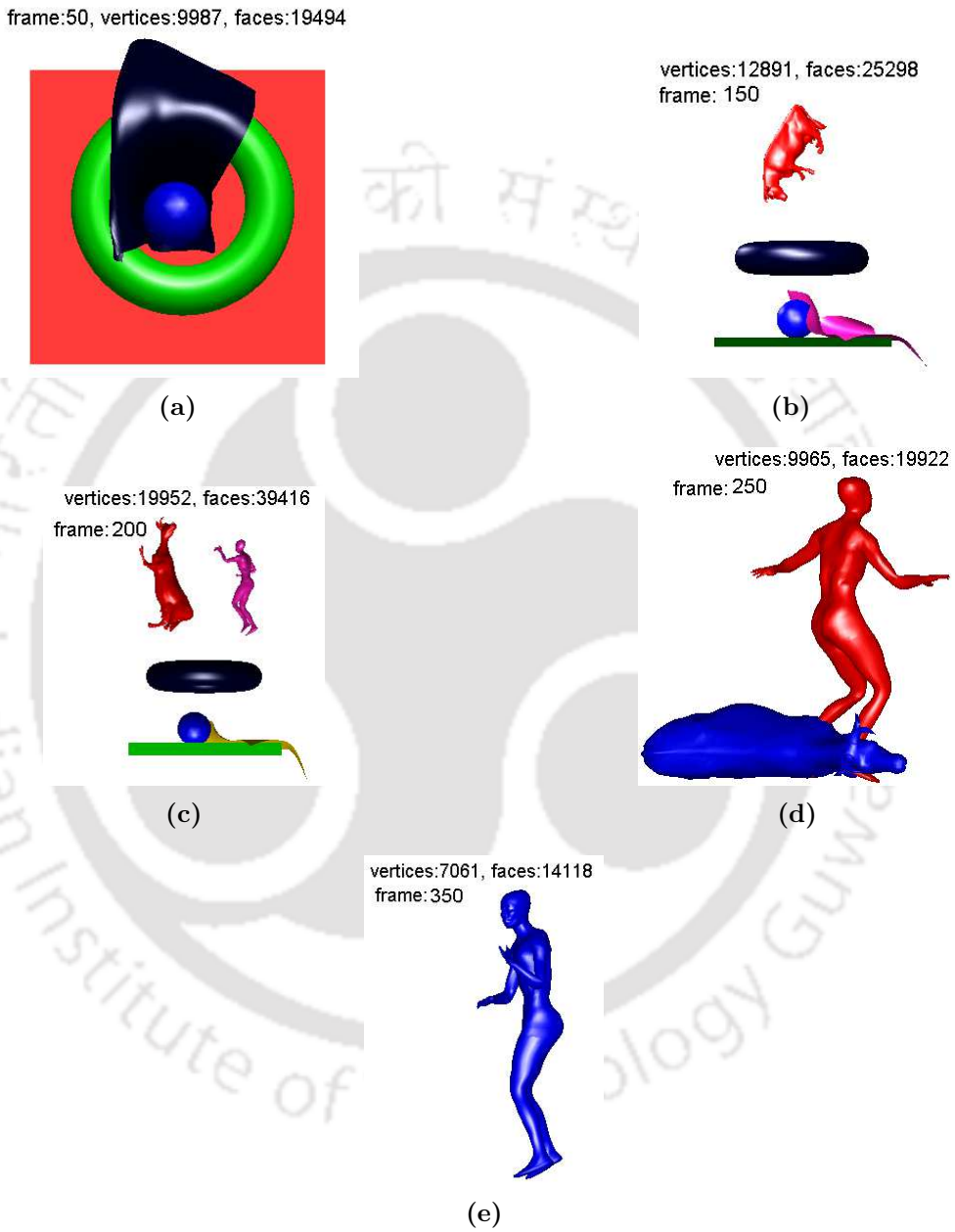


Figure 4.13: Segmentation of 3D objects in five IMO segments for "Cloth-Cow-Dance" BIMO-3D animation: (a) 4 objects in 1st IMO segment, (b) 5 objects in 2nd IMO segment, (c) 6 objects in 3rd IMO segment, (d) 2 objects in 4th IMO segment and (e) 1 object in 5th IMO segment

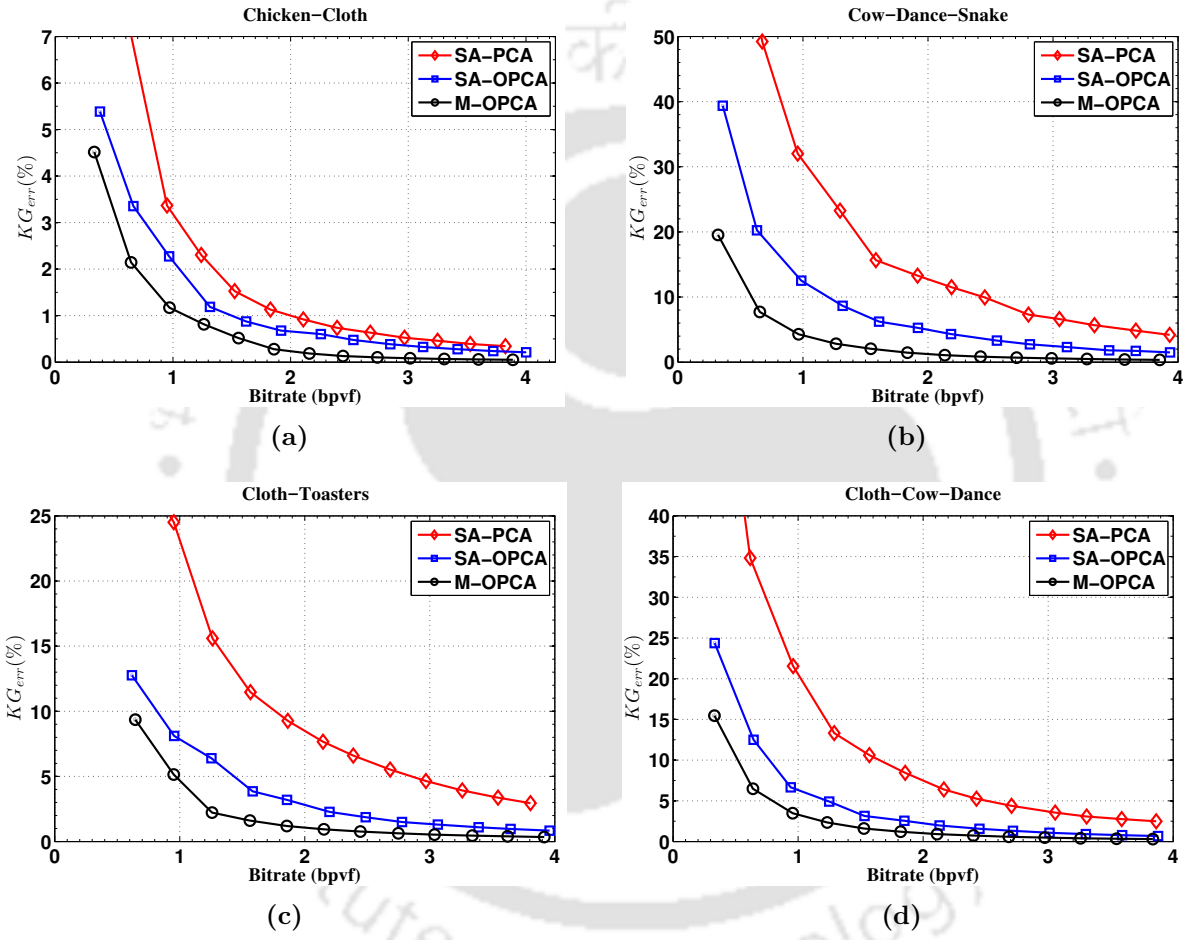


Figure 4.14: Comparative performance of the SA-PCA, the SA-OPCA and the M-OPCA algorithms in terms of KG_{err} metric on BIMO-3D animations: (a) 'Chicken-Cloth" sequence with 5 objects in 3 IMO blocks, (b) "Cow-Dance-Snake" sequence with 3 objects in 3 IMO blocks, (c) "Cloth-Toasters" sequence with 45 objects in 3 IMO bloks and (d) "Cloth-Cow-Dance" sequence with 6 objects in 3 IMO blocks.

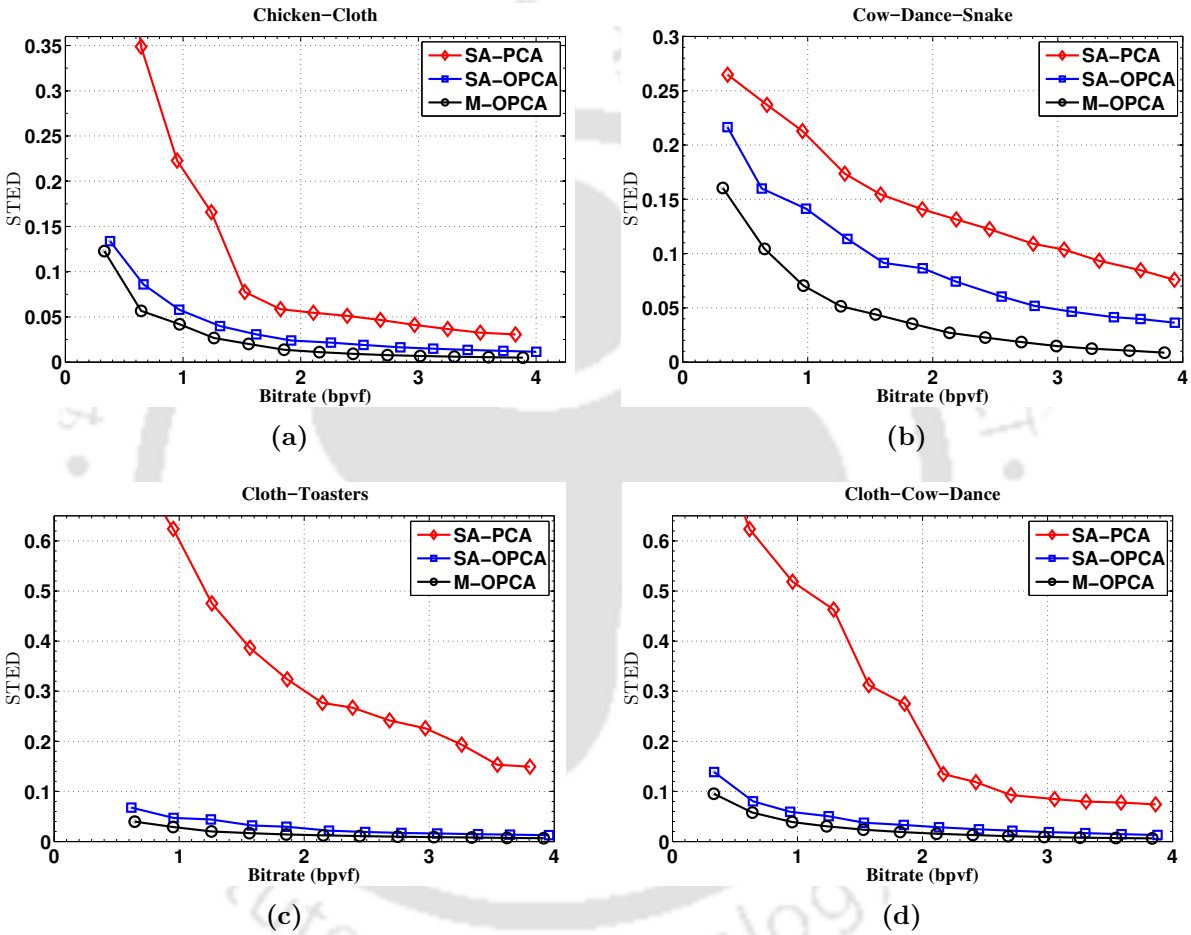


Figure 4.15: Comparative performance of the SA-PCA, the SA-OPCA and the M-OPCA algorithms in terms of the *STED* metric on BIMO-3D animations: (a) "Chicken-Cloth" sequence with 5 objects in 3 IMO blocks, (b) "Cow-Dance-Snake" sequence with 3 objects in 3 IMO blocks, (c) "Cloth-Toasters" sequence with 45 objects in 3 IMO blocks, and (d) "Cloth-Cow-Dance" sequence with 6 objects in 3 IMO blocks.

4.7 Conclusions

This chapter presented a novel method for the geometry compression of multi-object 3D animations. It exploits the spatio-temporal correlations of the individual 3D objects across the animation duration. The method deals with compression of geometry data of BIMO-3D animations. The BIMO-3D animations contain multiple 3D objects per frame and satisfy the graph isomorphism property in sets of successive frames. The proposed method applies a temporal segmentation algorithm to divide the animation data of the consecutive frames into different IMO blocks. Each IMO block satisfies the graph isomorphism property for the set of frames belonging to it. The proposed connectivity based spatial segmentation algorithm is applied on each IMO block to detect the vertices belonging to the individual 3D objects. The geometry data of the individual 3D object are then separated out for each IMO block. The 3D objects spanning across multiple IMO blocks are detected based on their mesh attributes and Euclidean distances. The geometry data of those unique 3D objects are extracted across the frames in IMO blocks. The OPCA algorithm is finally applied on the vertex trajectories of the unique 3D objects to get the compression. The compression performance of the proposed merged OPCA (M-OPCA) method is evaluated on a few BIMO-3D animations. The performance has been compared with the SA-PCA and SA-OPCA methods which apply the PCA and OPCA algorithm respectively on geometry data of each IMO block without any merging operation. For the "Cloth-Cow-Dance" sequence, at $bpvf \approx 2.0$, the KG_{err} value is 0.9098 for M-OPCA, whereas for SA-OPCA it is 1.97. For SA-PCA, the error is high at 6.39. For other sequences also, the merged OPCA is found to be better than that of the SA-PCA and SA-OPCA methods. Thus, the proposed method is suitable for 3D animation compression having multiple objects in sequential frames.

5

An Improved Distortion Metric for 3D Animation Geometry

Contents

5.1 Introduction

The dynamic 3D meshes undergo various processing operations for reliable storage, transmission, security and visualization. These operations include compression algorithms for efficient storage and streaming through network environments, watermarking for security and copyright protection, level-of-details (LOD) simplification for faster processing and rendering. Most of these operations are applied mainly on the geometry part of the animation data. Geometric distortions are introduced in the original data because of these operations. The geometrical distortions result in visual artifacts in the reconstructed animation after rendering and alter the perceptual quality of the 3D content with respect to the original one. Hence, the quality assessment of 3D animation is very much essential. The quality of the reconstructed animation can be evaluated using a *quality metric*. This may also help for the development and optimization of the geometry processing algorithms. A quality metric is alternatively termed as a Distortion metric (DM). A DM signifies the amount of distortion present or visible in a reconstructed 3D animation. The score of a DM can be correlated with the quality score because a high quality metric value signifies a low amount of distortion introduced. This chapter investigates the existing DMs for 3D animations. An improved DM is proposed and its performance is compared with other existing metrics in terms of correlation with the subjective quality scores.

The rest of the chapter is organised as follows: Section 5.2 presents an overview of different DMs used for the quality assessment of static and dynamic 3D meshes. Section 5.3 presents the details about the existing perceptual and non-perceptual DMs used specifically for 3D animations. The details of the proposed perceptual visual distortion (PVD) metric designed for 3D animations is presented in Section 5.4. Section 5.5 presents the performance evaluation of the proposed PVD metric in terms of correlation with subjective scores available for two publicly available 3D animation databases. Section 5.6 concludes the chapter by discussing the benefits of the proposed PVD metric.

5.2 Distortion metrics for 3D meshes

To evaluate the performance of geometry compression algorithms, it is necessary to perform the quality assessment of the decompressed animation data. The quality assessment involves the *subjective* or *objective* evaluation of the reconstructed 3D mesh data with respect to its original. The subjective evaluation refers to the quality assessment of the reconstructed or impaired 3D meshes by human

observers in terms of opinion scores based on the visual quality. The subjective opinion scores for a 3D mesh can be expressed in terms of either *mean opinion score* (MOS) value or *differential mean opinion score* (DMOS) value [71]. The MOS value of a distorted 3D mesh is obtained by averaging the raw opinion scores from different human observers for that mesh. The raw opinion scores are generally obtained as integer values with a scale of either from 0 to 10 or from 0 to 5. The DMOS value of a distorted 3D mesh is obtained by averaging the difference scores computed by subtracting the raw opinion scores of the distorted mesh from the reference score of the corresponding original mesh. However, carrying out subjective assessment is always time consuming, which cannot be made as an automatic process. The researchers have, therefore, focused on the development of objective DMs which can automatically predict the quality. An objective DM can be easily deployed along with the mesh compression algorithms to assess and control the visual quality of the final outputs.

The objective DMs for 3D meshes are computed by two approaches: *model-based* and *image-based* approaches [72]. The model-based approach computes the DM from the geometrical or other structural information of the 3D models in the 3D object space. These DMs are also termed as *viewpoint-independent* metrics. In the 3D space, the 3D mesh geometry can be viewed from any viewpoint [73]. On the other hand, the image-based approach computes the DM values in the image space using the images of the 3D models rendered from a viewpoint. The image-based DMs are also termed as *viewpoint-dependent* metrics since different images are obtained for a 3D model when it is rendered from different viewpoints. The rendered images depend on the surface colour, texture information of the 3D models and also on the lighting condition at the viewpoint.

The goal of any model-based or image-based DM is to predict the quality of a distorted 3D mesh in an automatic manner correlating with the subjective score. However, not all of the model-based or image-based DMs applied on the distorted meshes can give outputs in correlation with subjective scores. Hence, the researchers in the computer graphics domain have started to integrate the properties of the HVS with the objective DMs. Based on the nature of integration of the properties of the *human visual system* (HVS), the DMs for 3D meshes can be classified into two categories: *non-perceptual* and *perceptual* DMs [74]. The non-perceptual DMs are directly computed from the geometric information of the 3D meshes without taking into account of any correlation effects of the HVS. On the other hand, the perceptual DMs depend on the geometric information, texture information and different

psycho-perceptual factors related to HVS. They are computed by integrating the properties of the HVS like sensitivity to contrast, colour, visual masking and so on. The perceptual DMs perceive the visual quality of the distorted 3D meshes better than the non-perceptual DMs as reported in the literature [74], [75], [73], [71], [76].

Image-based DMs for 3D Meshes

The image-based approach considers the rendered image of the 3D model taken from a particular viewpoint and it provides a view-dependent quality assessment of 3D meshes. A number of image-based DMs for static 3D meshes have been developed by applying the image processing algorithms on the rendered images of 3D meshes as done in the computation of 2D image metrics [77], [78], [79], [80], [81], [82], [83], [84], [74], [73], [85]. However, there are problems with the image-based DMs in the context of computer graphics applications as reported by Rogowitz and Rushmeier in [86]. It was demonstrated that the perceived quality measured by the image-based DMs from the 2D images of 3D meshes are not adequate to evaluate and correctly predict the perceived quality compared to the model-based DMs in the geometric space. They have made the following observations:

(a) The scene lighting conditions strongly affect the perceptual image-based DMs compared to the geometric metric (model-based); and

(b) For 3D meshes undergoing simplification operation, the quality of the 3D objects perceived by the human observers differs according to whether the objects are observed as static 2D projected images or as an animated sequence of the 2D projected images taken from different viewpoints. The difference in perceived quality is difficult to integrate in the perceptual image-based DMs.

For the above two main disadvantages, the image-based DMs are not considered in our investigation. In this chapter, we have consider only the model-based DMs for the visual quality assessment of 3D meshes. An overview of the model-based DMs that are most commonly used for quality measurement of 3D meshes from various processing algorithms is presented below.

Model-based Objective DMs for 3D Meshes

The model-based objective DMs are initially applied in the context of quality assessment of 3D meshes processed using the simplification algorithms. The goal was to control the level of mesh

simplification steps to represent the 3D mesh with reduced number of vertices while preserving the visual quality. Most of the existing model-based DMs for 3D meshes were originally developed for the static 3D meshes. The model based DMs for 3D meshes are also divided into *non-perceptual* and *perceptual* types based on the integration of characteristics of HVS in computing the metric values. Some of the popular and widely used model-based non-perceptual DMs for static 3D meshes are the root mean square (RMS) error, the Metro distortion metric using the Hausdorff distance (HD) [87], the maximum root mean square (MRMS) error [88] and the peak signal to noise ratio (PSNR) [61], [89].The above model-based non-perceptual DMs fail to correctly capture the visual quality difference between two 3D meshes correlating with the perceived visual quality of human observers. These simple metrics provide a very poor predictor of visual fidelity (as demonstrated in [72]), since they are computed based on the geometrical distances without taking into account of the characteristics of the HVS. To address this, a number of model-based perceptual visual distortion (PVD) metrics have been developed by the researchers in the computer graphics domain. These PVD metrics integrate different perceptual characteristics of the HVS such as the sensitivities to curvature information, edge lengths, dihedral angles, surface normals and texture information. The texture mapping on the triangular meshes is important for rendering the visual quality of 3D animated meshes to provide realistic visualization. The references [107], [108], [109] and [110] consider geometry, texture and other perceptual factors for developing perceptual DMs. Some of the existing model-based PVD metrics for static 3D meshes are the geometric Laplacian (GL) [17], the mesh structural distortion measure (MSDM) [90], [91], the dihedral angle mesh error (DAME) [92], the fast mesh perceptual distortion (FMPD) [93] and the tensor-based perceptual distortion measure (TPDM) [94].

5.3 Model-based DMs for 3D animations

A 3D animation comprises of a series of static mesh frames. The model-based DMs that are used for the evaluation of 3D animations can be applied in two different ways. (i) In the first approach, any DM developed for the static 3D meshes is applied in a frame-wise manner and the final DM value for the animation is given by the sum, average or maximum value of frame-wise DMs. (ii) In the second approach, the DM for a 3D animation is computed by considering the whole dynamic geometry data as a single entity in a matrix form. This class of DMs can address both the spatial and temporal artifacts

that occur in a 3D animation, which is not possible in the first approach. Like in the case of static 3D meshes, the existing model-based DMs for the 3D animations are also divided into *non-perceptual* and *perceptual* types based on their correlations with the HVS. An overview of the existing non-perceptual and perceptual model-based DMs for 3D animations is presented below.

5.3.1 Model-based non-perceptual DMs for 3D animations

Consider two 3D animations $\mathcal{M}_d$ and $\tilde{\mathcal{M}}_d$, each consisting of N_f frames and N_v number of vertices per frame with identical connectivity. Let $\mathbf{p}_{v_i}^f$ and $\tilde{\mathbf{p}}_{v_i}^f \in \mathbb{R}^3$ denote the coordinates of i-th vertex of f-th frame of $\mathcal{M}_d$ and $\tilde{\mathcal{M}}_d$ respectively. The following are the non-perceptual DMs for these meshes:

(i) **Average root mean square (ARMS) error**

The ARMS error between $\mathcal{M}_d$ and $\tilde{\mathcal{M}}_d$ is computed as the average of the frame-wise RMS difference calculated over all the frames and written as

$$ARMS(\mathcal{M}_d, \tilde{\mathcal{M}}_d) = \frac{1}{N_f}\sum_{f=1}^{N_f}\sqrt{\frac{1}{N_v}\sum_{i=1}^{N_v}\left\|\mathbf{p}_{v_i}^f - \tilde{\mathbf{p}}_{v_i}^f\right\|^2} \tag{5.1}$$

(ii) **Average Hausdorff distance (AHD)**

The Hausdorff distance (HD) metric has been applied by Cho et al. [25] to evaluate the performance of the wavelet based compression of 3D animations. The average Hausdorff distance (AHD) between $\mathcal{M}_d$ and $\tilde{\mathcal{M}}_d$ is computed as the average of the frame-wise symmetric HDs between the original and the reconstructed mesh sequences and written as

$$AHD(\mathcal{M}_d, \tilde{\mathcal{M}}_d) = \frac{1}{N_f}\sum_{i=1}^{N_f}\max\{d_H(\mathcal{M}_{s_i}, \tilde{\mathcal{M}}_{s_i}), d_H(\tilde{\mathcal{M}}_{s_i}, \mathcal{M}_{s_i})\} \tag{5.2}$$

where $d_H(\mathcal{M}_{s_i}, \tilde{\mathcal{M}}_{s_i})$ is the directed one-sided Hausdorff distance between two static mesh surfaces $\mathcal{M}_{s_i}$ and $\tilde{\mathcal{M}}_{s_i}$ with the number of vertices N_v and N_v' respectively and defined as

$$d_H(\mathcal{M}_{s_i}, \tilde{\mathcal{M}}_{s_i}) = \max_{\forall v_j \in \mathcal{M}_{s_i}} d(v_j, \tilde{\mathcal{M}}_{s_i}), \tag{5.3}$$

The $d(v_j, \tilde{\mathcal{M}}_{s_i})$ is the distance between a vertex v_j on the mesh $\mathcal{M}_{s_i}$ to a closest vertex v_k' on the mesh $\tilde{\mathcal{M}}_{s_i}$ and given by

$$d(v_j, \tilde{\mathcal{M}}_{s_i}) = \min_{\forall v_k' \in \tilde{\mathcal{M}}_{s_i}}\left\|\mathbf{p}_{v_j}^i - \mathbf{p}_{v_k'}^i\right\|, j \in 1 \ldots N_v, \quad k \in 1 \ldots N_v' \tag{5.4}$$

We can similarly define $d_H(\tilde{\mathcal{M}}_{s_i}, \mathcal{M}_{s_i})$.

(iii) **Average maximum root mean square (AMRMS) error**

The AMRMS error is also an HD based DM and it was used in [95] for evaluating the coding performance of dynamic 3D meshes obtained using the spatial and temporal wavelet analysis. The AMRMS error between $\mathcal{M}_d$ and $\tilde{\mathcal{M}}_d$ is computed as the average of the frame-wise MRMS error and written as

$$AMRMS(\mathcal{M}_d, \tilde{\mathcal{M}}_d) = \frac{1}{N_f}\sum_{i=1}^{N_f}\frac{1}{D}\max\{d_{RMSE}(\mathcal{M}_{s_i}, \tilde{\mathcal{M}}_{s_i}), d_{RMSE}(\tilde{\mathcal{M}}_{s_i}, \tilde{\mathcal{M}}_s)\} \tag{5.5}$$

The frame-wise MRMS error is computed as the maximum of the two directed RMS error distances between the two surfaces and defined as

$$MRMS(\mathcal{M}_s, \tilde{\mathcal{M}}_s) = \frac{1}{D}\max\{d_{RMSE}(\mathcal{M}_s, \tilde{\mathcal{M}}_s), d_{RMSE}(\tilde{\mathcal{M}}_s, \mathcal{M}_s)\}, \tag{5.6}$$

where $d_{RMSE}(\mathcal{M}_{s_i}, \tilde{\mathcal{M}}_{s_i})$ is the directed RMS error distance from $\mathcal{M}_s$ to $\tilde{\mathcal{M}}_s$ and is defined as

$$d_{RMSE}(\mathcal{M}_s, \tilde{\mathcal{M}}_s) = \sqrt{\frac{1}{|\mathcal{M}_s|}\sum_{v_i \in \mathcal{M}_s} d(v_i, \tilde{\mathcal{M}}_s)^2} \tag{5.7}$$

Similarly, the directed RMS error $d_{RMSE}(\tilde{\mathcal{M}}_s, \mathcal{M}_s)$ from $\tilde{\mathcal{M}}_s$ to $\mathcal{M}_s$ is computed. The calculation of both AHD and $AMRMSE$ metric values for dynamic 3D meshes requires high computation time and large memory.

(iv) **Mean square error (MSE)**

A full reference mean square error (MSE) based DM for the dynamic 3D meshes was first proposed by Heu et al. [96]. The MSE between $\mathcal{M}_d$ and $\tilde{\mathcal{M}}_d$ is computed by considering the whole animation geometry data in a matrix form and given by

$$MSE(\mathcal{M}_d, \tilde{\mathcal{M}}_d) = \frac{\left\|\mathbf{A} - \widetilde{\mathbf{A}}\right\|_f^2}{3N_v N_f} \tag{5.8}$$

where $\mathbf{A}$ and $\widetilde{\mathbf{A}}$ are matrices representing the geometry data of $\mathcal{M}_d$ and $\tilde{\mathcal{M}}_d$ respectively with dimension $[3N_v \times N_f]$ and the notation $\|.\|_f$ denotes the Frobenius norm. The computation time of this metric is lower than that of the AHD and AMRMS metrics.

(v) **Signal to noise ratio (SNR)**

A objective DM based on the SNR was used by Yang et al. [97] to evaluate the performance of their prediction based 3D animation compression algorithm. It is defined as

$$SNR(\mathcal{M}_d, \tilde{\mathcal{M}}_d) = 10 \log_{10} \frac{\sum_{f=1}^{N_f} \sum_{i=1}^{N_v} \left\| \mathbf{p}_{v_i}^f \right\|^2}{\sum_{f=1}^{N_f} \sum_{i=1}^{N_v} \left\| \mathbf{p}_{v_i}^f - \tilde{\mathbf{p}}_{v_i}^f \right\|^2} \tag{5.9}$$

(vi) **KG-error**

Karni and Gotsman [3] proposed a full reference model-based DM for 3D animations. It is abbreviated as KG_{err} in the literatures and based on the pure geometric distance between meshes of 3D animations. It is the most widely used DM for the 3D animations and defined as

$$KG_{err}(\mathcal{M}_d, \tilde{\mathcal{M}}_d) = \frac{\left\| \mathbf{A} - \tilde{\mathbf{A}} \right\|_f}{\left\| \mathbf{A} - \mathbf{E}(\mathbf{A}) \right\|_f} \times 100\% \tag{5.10}$$

where $\mathbf{E}(\mathbf{A})$ is the mean matrix of size $[3N_v \times N_f]$ with each column consisting of the mean frame of the animation computed from matrix $\mathbf{A}$ by averaging its N_f columns. The KG_{err} approach is different from the MSE calculation as it takes a different normalization parameter. The denominator term ensures that the value of KG_{err} is invariant to the uniform scaling of both the original and reconstructed datasets. However, it is not the best distortion metric for animated meshes as it is not invariant to global translation or rotation of the dataset.

The above model-based DMs have been used by various researchers for the quality evaluation of the dynamic 3D mesh processing algorithms. However, these DMs lack correlation with the subjective assessment scores for the visual quality. Moreover, the most of the above DMs do not show invariance to global scaling, translation or rotation effects of the animated meshes. To address the above issues, researches have started developing model-based PVD metrics for 3D animations by incorporating the properties of the HVS to get the maximum correlation with the subjective opinion scores. An overview of the existing PVD metrics for 3D animations is presented below.

5.3.2 Model-based PVD metrics for 3D animations

The model-based PVD metrics for 3D animations have not been explored much. Some of the PVD metrics developed for the static 3D meshes can be extended to detect visual distortion in a 3D

animation using the average or maximum value of the frame-wise DMs. However, the application of the PVD metrics for static 3D meshes on the 3D animations cannot detect temporal artifacts which may occur across the frames. They also cannot address the temporal visual masking effect sensitive to the HVS, since they do not take into account the temporal features related to the speed and the motion directions of the mesh vertices. Hence, the development of a model-based PVD metrics by considering both the spatial and temporal features of 3D animations is an important research area. A PVD metric should always give output values correlating with the HVS. To verify that, it is necessary to evaluate the correlation performance of a PVD metric with the subjective opinion scores given by human observers based on the visual quality of distorted 3D animations.

5.3.2.1 Measure of correlations between the PVD metric and subjective scores

The correlation performances of a PVD metric are measured in terms of *Pearson's linear correlation coefficient* (PLCC) and *Spearman's rank order correlation coefficient* (SROCC) [49]. The PLCC measures the linear correlation between the objective metric and the subjective scores to provide the prediction accuracy of the metric. The SROCC value measures the statistical dependence between the ranks of objective and subjective scores to give the prediction monotonicity of the metric.

Consider two data sets X_o and Y_s with N_s number of samples of the objective metrics x_i and their corresponding subjective scores y_i as the elements. The PLCC value ρ_p between the two sets X_o and Y_s is computed as

$$\rho_p = \frac{\sum_i (x_i - \bar{x}_o)(y_i - \bar{y}_s)}{\sqrt{\sum_i (x_i - \bar{x}_o)^2 (y_i - \bar{y}_s)^2}} \tag{5.11}$$

where $\bar{x}_o$ denotes mean value of the objective metrics and $\bar{y}_s$ denotes mean value of the subjective scores expressed in terms of either a MOS or a DMOS value. The value of ρ_p lies in the interval -1 to 1, where a $\rho_p = +1$ signifies linear dependence, $\rho_p = -1$ signifies negative linear dependence and $\rho_p = 0$ signifies zero correlation between the values of X_o and X_s.

The SROCC provides a measure of linear association between the ranks of the elements of X_o and Y_s. Let r_{x_i} and r_{s_i} represent the ranks of the objective metrics x_i and the corresponding subjective scores y_i. Each rank is an integer value, from 1 to N_s. The values of objective metric, x_is, are ranked from low to high such that rank 1 indicates the lowest value and rank N_s indicates the highest value.

The subjective scores, y_is, are also ranked from low to high such that rank 1 is assigned to the lowest DMOS value and rank N_s corresponds to the highest DMOS value. The SROCC ρ_s is obtained by using the ranks values, r_{x_i} and r_{s_i}, in the computation of PLCC and given as

$$\rho_s = \frac{\sum_i (r_{x_i} - \bar{r}_x)(r_{y_i} - \bar{r}_y)}{\sqrt{\sum_i (r_{x_i} - \bar{r}_x)^2 (r_{y_i} - \bar{r}_y)^2}} \tag{5.12}$$

where $\bar{r}_x$ denotes mean value of r_{x_i} and $\bar{r}_y$ denotes mean value of r_{y_i}. Like the PLCC ρ_p, the value of ρ_s lies in the interval -1 to 1, where $\rho_s = +1$ signifies perfect positive correlation, $\rho_s = -1$ signifies perfect negative correlation and $\rho_s = 0$ signifies zero correlation between the rank values of x_i and y_i.

The existing quality assessment methods have recommended a step of curve fitting between the subjective scores and the objective metrics before computing the PLCCs. This curve fitting is done using a suitable non-linear psychometric function [98], [99], [71] which maps the objective metric values to the subjective scores represented by the MOS or DMOS values. The use of the non-linear psychometric function helps in removing any non-linearity from the subjective scores. The most commonly used non-linear psychometric function is the three-parameter *logistic function* given as

$$f(a, b, c, m) = \frac{a}{1 + e^{-b(m-c)}} \tag{5.13}$$

where m is the actual objective metric before fitting and the parameters a, b and c are obtained through a non-linear least squares fitting between the actual objective metrics and the corresponding MOS or DMOS values from the subjective assessments.

5.3.3 Existing model-based PVD metrics

Vasa et al. [9] have proposed the first full-reference PVD metric for the 3D animations called the *spatio-temporal edge difference* (STED) metric for evaluating the performance of geometry compression algorithms. The STED metric is based on the fact that the perception of distortion is related to the local and the relative changes of edge lengths rather than the global and absolute changes of the vertex geometry. This metric uses the edge attribute of the dynamic 3D mesh to compute the relative changes in the spatial and the virtual temporal edge lengths between the original and the distorted mesh sequences across the animation. The virtual temporal edges are formed by connecting a vertex in a frame to its positions in the subsequent frames. The final metric value is given by the hypotenuse

of right-angle triangle represented by the spatial and temporal edge lengths errors across the frame.

Another PVD metric for 3D animations called the *dynamic mesh perceptual distance* (DMPD) was proposed by Torkhani et al. [71]. The metric is computed as the weighted mean square combinations of three perceptual measures correlated with the HVS. The first one is the speed-weighted surface local roughness distortion measure, the second one is the vertex motion based distortion measure and the third one is the vertex motion direction based distortion measure. A database was also created by the authors by applying various distortions on some 3D animation sequences including the spatial and visual masking effects. We present below an outline of the STED [9] and the DMPD [71] metrics specifically developed for the quality assessment of 3D animations.

(a) **STED metric**

As discussed above, the STED metric combines the spatial and the temporal deviations of the edge lengths computed at each vertex of the original and the distorted frames of the animations. The spatial part computes the standard deviation of the relative edge lengths difference between the original and the distorted animations at each vertex of the frames within its topological neighbourhood vertices. Thus, it focusses mainly on the local change of errors. These relative edge differences for the vertices are summed over all the frames to get the overall spatial part of the STED metric. The temporal part computes the difference between the virtual temporal edges in the original and the distorted meshes for each vertex of the frames. The temporal edge differences are also normalized by the speed of the vertex within a local temporal windows to take into account the shaking artifacts which are more visible in the areas that move slowly or remain static during the animation. The mathematical details for computing the spatial and temporal parts of the STED metric between two animations, $\mathcal{M}_d$ and $\tilde{\mathcal{M}}_d$, are explained below.

(i) **Spatial edge error**

Let e_{ij} represent a spatial edge connecting two vertices v_i^f and v_j^f in the frame f of the original animation $\mathcal{M}_d$ and $\tilde{e}_{ij}$ be the corresponding spatial edge in the frame f of the reconstructed animation $\tilde{\mathcal{M}}_d$. The lengths of the spatial edges e_{ij} and $\tilde{e}_{ij}$ in the frame f of

$\mathcal{M}_d$ and $\tilde{\mathcal{M}}_d$ respectively are measured as

$$\begin{aligned} el(e_{ij}, f) &= \left\| \mathbf{p}_{v_i}^f - \mathbf{p}_{v_j}^f \right\| \\ el(\tilde{e}_{ij}, f) &= \left\| \tilde{\mathbf{p}}_{v_i}^f - \tilde{\mathbf{p}}_{v_j}^f \right\| \end{aligned} \tag{5.14}$$

where the notation $\|.\|$ denotes the Euclidean norm. The relative edge difference, $ed(e_{ij}, f)$, between the spatial edges e_{ij} and $\tilde{e}_{ij}$ for the f-th frame is now measured as

$$ed(e_{ij}, f) = \left\| \frac{el(e_{ij}, f) - el(\tilde{e}_{ij}, f)}{el(e_{ij}, f)} \right\| \tag{5.15}$$

Note that the $ed(e_{ij}, f)$ measure increases the sensitivity to the distortion that occurs in the densely populated areas of a mesh frame. This relative edge difference is measured for the edges formed by each vertex v_i^f with its topological neighbourhood vertices for the user specified topological distance d. The set of vertices with topological distance d signifies the set of vertices which are apart up to a maximum d edges from the i-th vertex and is denoted as $n_v(i, d)$. Accordingly, a set of neighbouring edges, $n_e(i, d)$, incident on each of the vertex in the set $n_v(i, d)$ is obtained. The lengths of edges in the set $n_e(i, d)$ are different for each vertex v_i^f. So, the average relative edge difference $ed_{avg}(i, f, d)$ around v_i^f is computed as the weighted average of the relative edge difference with the neighbourhood edges and given by

$$ed_{avg}(i, f, d) = \frac{\sum\limits_{e \in n_e(i,d)} ed(e, f) el(e, f)}{\sum\limits_{e \in n_e(i,d)} el(e, f)} \tag{5.16}$$

The weight of each edge is determined by its length in the original mesh frame. Now, the local deviation of the edge differences $dev(i, f, d)$ around v_i^f for the topological distance d is computed as

$$dev(i, f, d) = \sqrt{\frac{\sum\limits_{e \in n_e(i,d)} (ed(e, f) - ed_{avg}(i, f, d))^2 el(e, f)}{\sum\limits_{e \in N_e(i,d)} el(e, f)}} \tag{5.17}$$

The average of the local deviations over N_v vertices and N_f frames of the mesh sequence contributes to the spatial error part of the STED metric for the given distance d and is

given by

$$STED_s(d) = \frac{1}{N_v N_f} \sum_{i=1}^{N_v} \sum_{j=1}^{N_f} dev(i, f, d) \tag{5.18}$$

(ii) **Temporal edge error**

The calculation of the temporal edge error is similar to that of the spatial edge error with the exception that the virtual edges connecting the vertices in the subsequent frames are considered in this case. The temporal edge length $tel(i, f)$ between the vertex v_i^f in a frame f and the corresponding vertex v_i^{f+1} in the frame $(f + 1)$ of the original animation $\mathcal{M}_d$ is calculated as follows:

$$\begin{aligned} ld &= \max_{i,j \in \{1 \ldots N_v\}} \left(\left\| \mathbf{p}_{v_i}^1 - \mathbf{p}_{v_j}^1 \right\| \right) \\ dx(i, f) &= \frac{v_{i_x}^{f+1} - v_{i_x}^f}{ld} \\ dy(i, f) &= \frac{v_{i_y}^{f+1} - v_{i_y}^f}{ld} \\ dz(i, f) &= \frac{v_{i_z}^{f+1} - v_{i_z}^f}{ld} \\ tel(i, f) &= \sqrt{dx(i, f)^2 + dy(i, f)^2 + dz(i, f)^2 + d_t^2} \end{aligned} \tag{5.19}$$

where $v_{i_x}^f$, $v_{i_y}^f$ and $v_{i_z}^f$ are the x, y and z coordinates of the vertex v_i^f and d_t is an empirical constant used to avoid zero length temporal edges from the processing of static vertices. Similarly, the temporal edge lengths $\tilde{tel}(i, f)$ for the vertices in the reconstructed animation $\tilde{\mathcal{M}}_d$ are computed. The values of $tel(i, f)$ and $\tilde{tel}(i, f)$ are not defined for the vertices in the last frame N_f.

To evaluate the steadiness of the movement of the vertex v_i^f in a frame f, the average spatio-temporal speed s_t of the vertex is computed within a temporal window of size w around f as

$$s_t(i, f, w) = \frac{\sum_{lf=\max(1, f-w)}^{\min(N_f, f+w)} tel(i, lf)}{\min(N_f, f + w) - \max(1, f - w)}. \tag{5.20}$$

The use of the relative temporal edge length instead of the absolute length increases the sensitivity of the metric in the areas of very slow motion. The relative temporal edge

difference for the i-th vertex in the original and reconstructed frame f can be defined as

$$ted(i,f,w) = \frac{|tel(i,f) - \tilde{tel}(i,f)|}{s_t(i,f,w)}. \tag{5.21}$$

Finally, the overall temporal error is calculated as the average over all the N_v vertices and $(N_f - 1)$ frames and given by

$$STED_t(w) = \frac{1}{N_v(N_f-1)} \sum_{i=1}^{N_v} \sum_{f=1}^{N_f-1} ted(i,f,w). \tag{5.22}$$

The overall spatio-temporal edge difference, $STED$, is now calculated by combing the $STED_s$ and $STED_t$ as

$$STED(d,w) = \sqrt{STED_s(d)^2 + c^2.STED_t(w)^2}. \tag{5.23}$$

where c is the weighting coefficient parameter.

From the above expression, it is clear that the STED metric depends on several parameters such as the width of the topological neighbourhood d, width of temporal local windows w, the distance between two consecutive frames d_t and the coefficient c for weighting the spatial and the temporal part. As per the experimental results mentioned in [49], the values for these $STED$ parameters were set as $d = 1$, $w = 5$, $d_t = 0.0003$ and $c = 9.144 \times 10^{-5}$ to get highest possible correlation with the subjective opinion scores.

(b) **DMPD Metric**

The DMPD metric between two dynamic 3D meshes $\mathcal{M}_d$ and $\tilde{\mathcal{M}}_d$ is computed as the weighted mean of three perceptual distances called the *speed-weighted spatial-based perceptual distance* (SWSPD), the *speed based perceptual distance* (SPPD) and the *angle-based perceptual distance* (APD) and written as

$$DMPD(\mathcal{M}_d, \tilde{\mathcal{M}}_d) = \sqrt{w_1(SWSPD)^2 + w_2(SPPD)^2 + w_3(APD)^2} \tag{5.24}$$

where w_1,w_2 and w_3 are the non-negative weights such that $w_1 + w_2 + w_3 = 1$. The steps for computing the $SWSPD$, the $SPPD$ and the APD are discussed below.

(i) **Speed-weighted spatial-based perceptual distance (SWSPD)**

The SWSPD metric between $\mathcal{M}_d$ and $\tilde{\mathcal{M}}_d$ employs the spatial perceptual features such as the local roughness of the mesh surface for calculating the perceptual distance. Let $n_f(i)$ be the set of all the neighbouring facets at vertex v_i^f of the frame f in $\mathcal{M}_d$ and $n_v(i,1)$ is the set of all the neighbouring vertices of v_i^f with topological distance $d = 1$. The value of the discrete Gaussian curvature GC_i^f at v_i^f is then computed as

$$GC_i^f = \left| 2\pi - \sum_{k \in n_f(i)} \theta_k \right| \tag{5.25}$$

where θ_k is the angle made by the facet $k \in n_f(i)$ with v_i^f as shown in Figure 5.1(a).

Figure 5.1: (a) Angle θ_k formed by a facet to v_i^f. (b) Angles β_{ij}^f and $\hat{\beta}_{ij}^f$ for an edge connecting v_i^f and v_j^f.

Likewise, the mesh Laplacian matrix D_{ij}^f at v_i^f is computed using its neighbouring vertex $v_j^f, j \in n_v(i,1)$ and given by

$$\begin{aligned} D_{ij}^f &= \frac{\cot(\beta_{ij}^f) + \cot(\hat{\beta}_{ij}^f)}{2}, \quad i \neq j \\ \text{and} \quad D_{ii}^f &= -\sum_{j \in n_v(i,1)} D_{ij}^f \end{aligned} \tag{5.26}$$

where β_{ij}^f and $\hat{\beta}_{ij}^f$ are the two angles opposite to the edge connecting vertices v_i^f and v_j^f of frame f in $\mathcal{M}_d$ as shown in Figure 5.1(b). The local roughness value r_i^f at v_i^f is defined as the absolute value of the Laplacian of Gaussian curvature and given by

$$r_i^f = \left| GC_i{}^f - \frac{\sum_{j \in n_v(i)} D_{ij}^f GC_j^f}{\sum_{j \in n_v(i)} D_{ij}^f} \right| = \left| GC_i{}^f + \frac{\sum_{j \in n_v(i)} D_{ij}^f GC_j^f}{D_{ii}^f} \right|, \; i \in \{1, \ldots, N_v\}, \; f \in \{1, \ldots, N_f\} \tag{5.27}$$

Similarly, the local roughness value $\tilde{r}_i^f$ for each vertex $\tilde{v}_i^f$ of the frames in $\tilde{\mathcal{M}}_d$ is computed.

The difference of r_i^f and $\tilde{r}_i^f$ at each vertex v_i^f of the frames in $\mathcal{M}_d$ and $\tilde{\mathcal{M}}_d$ is computed as

$$\delta r(i,f) = \frac{\left| r_i^f - \tilde{r}_i^f \right|}{r_i^f + \tilde{r}_i^f + C_r}, \quad i \in \{1,\ldots,N_v\}, \; f \in \{1,\ldots,N_f\} \tag{5.28}$$

where $C_r = 0.002$ is a small constant to avoid numerical instability for the denominator value near zero [71]. $\delta r(i,f)$ contributes to the spatio-visual masking effect to some extent due to the terms r_i^f and $\tilde{r}_i^f$ in the denominator. Because, for the same amount of local roughness change as given by $\left| r_i^f - \tilde{r}_i^f \right|$, the value of $\delta r(i,f)$ will be smaller in a rougher region of the mesh (where r_i^f and $\tilde{r}_i^f$ have larger values) compared to that in a smoother region of the mesh.

The visibility of the spatial distortions is reduced to some extent in the fast moving regions of the dynamic meshes. To take into account this speeding effect, the $\delta r(i,f)$ values are weighted by the speed of the respective vertex in the original mesh to capture a hybrid spatio-temporal effect of the perceived distortions. The value of $\delta r(i,f)$ is weighted by a small weight if the vertex v_i^f moves very fast and vice-versa. The perceptual distance $\delta r_s(i,f)$ at each vertex v_i^f of the frames is computed as

$$\delta r_s(i,f) = \begin{cases} w_1^s \cdot \delta r(i,f) & \text{if } \; s_t(i,f) \le Th_1, \\ \left(w_1^s - \frac{w_1^s - w_2^s}{Th_2 - Th_1} \left[s_t(i,f) - Th_1 \right] \right) \cdot \delta r(i,f) & \text{if } \; Th_1 < s_t(i,f) \le Th_2, \\ w_2^s \cdot \delta r(i,f) & \text{if } \; s_t(i,f) > Th_2, \end{cases} \tag{5.29}$$

where $s_t(i,f)$ is the forward speed of the vertex v_i^f normalized by the scene bounding box diagonal of the dynamic mesh, Th_1 and Th_2 are the threshold constants for the vertex forward speed and w_1^s and w_2^s are the constants used for weighting the local roughness differences. The values of these constant parameters were set at $Th_1 = 0.01$, $Th_2 = 0.05$, $w_1^s = 1$ and $w_2^s = 0.3$ in the experiment performed in [71] for the improvement of overall correlation with the subjective scores.

The speed-weighted local perceptual distances $\delta r_s(i,f)$s for each vertex v_i^f are then spatially pooled together using the power-3 Minkowski sum to get the per-frame speed weighted local

perceptual errors $\delta r_s^t(f)$ as

$$\delta r_s^t(f) = \left(\frac{1}{N_v} \sum_{i=1}^{N_v} |\delta r_s(i,f)|^3 \right)^{\frac{1}{3}} \tag{5.30}$$

The values of $\delta r_s^t(f)$ for the N_f number of frames are sorted in an ascending order to get $\hat{\delta r}_s^t(f)$ such that $\hat{\delta r}_s^t(f_1) \leq \hat{\delta r}_s^t(f_2)$ for any $f_1 < f_2$. A set of weights $\hat{w}_f$ for all the N_f frames are then determined as

$$\hat{w}_f = \left(\frac{f}{N_f} \right)^{pw}, \tag{5.31}$$

where pw is a constant parameter to control the weights $\hat{w}_f$ along with the increase in frame number index f. A higher value of pw assigns higher weights to the frames with severe distortions. A value of $pw = 10$ was used for the experiments reported in [71] for improving the correlation between the objective and the subjective scores.

The final SWSPD metric between $\mathcal{M}_d$ and $\tilde{\mathcal{M}}_d$ is computed by pooling together the weighted combinations of per frame errors $\hat{\delta r}_s^t(f)$s using the weights $\hat{w}_f$ and given by

$$SWSPD(M_d, \tilde{M}_d) = \frac{\sum_{f=1}^{N_f} \hat{w}_f \cdot \hat{\delta r}_s^t(f)}{\sum_{f=1}^{N_f} \hat{w}_f} \tag{5.32}$$

(ii) **Speed based perceptual distance (SPPD)**

The SPPD metric between $\mathcal{M}_d$ and $\tilde{\mathcal{M}}_d$ is computed by incorporating the temporal feature related to the vertex speed. It is based on the difference in forward motion vectors calculated at each vertex of the frames of $\mathcal{M}_d$ and $\tilde{\mathcal{M}}_d$.

Let $\mathbf{d}_i^f$ and $\tilde{\mathbf{d}}_i^f$ represent the respective forward motion vectors of the vertices v_i^f and $\tilde{v}_i^f$ in the frame f of $\mathcal{M}_d$ and $\tilde{\mathcal{M}}_d$. These motion vectors are also normalized by the diagonal length of the scene bounding box to ensure scale invariance. For the vertices in the last frame of mesh sequences, the backward motion vectors have been used. A speed-based local perceptual distance $\delta s(i,f)$ between v_i^f and $\tilde{v}_i^f$ is computed using the difference in norm of

their motion vectors and given by

$$\delta s(i,f) = \frac{\left|\left\|\mathbf{d}_i^f\right\| - \left\|\tilde{\mathbf{d}}_i^f\right\|\right|}{\left\|\mathbf{d}_i^f\right\| + \left\|\tilde{\mathbf{d}}_i^f\right\| + C_s}, \quad i \in \{1,\ldots,N_v\}\,,\ f \in \{1,\ldots,N_f\} \tag{5.33}$$

where $C_s = 0.002$ is a constant used to avoid numerical instability. The values of $\delta s(i,f)$ also capture the temporal masking effect to some extent due to the terms $\left\|\mathbf{d}_i^f\right\|$ and $\left\|\tilde{\mathbf{d}}_i^f\right\|$ in the denominator. Because, the same value of speed change as given by $\left|\left\|\mathbf{d}_i^f\right\| - \left\|\tilde{\mathbf{d}}_i^f\right\|\right|$, will induce a smaller error value of $\delta s(i,f)$ in a fast-moving region of the mesh (i.e., where $\left\|\mathbf{d}_i^f\right\|$ and $\left\|\tilde{\mathbf{d}}_i^f\right\|$ values are higher) compared to that in a slowly moving or static part of the mesh. This is based on the observation that motion distortions are more difficult to perceive in a fast moving object than in a slowly moving object. The $\delta s(i,f)$ values for all the vertices in each frame are spatially pooled via the power-5 Minkowski sum to get the frame-wise local perceptual distances $\delta s^t(f)$ as given by

$$\delta s^t(f) = \left(\frac{1}{N_v}\sum_{i=1}^{N_v} |\delta s(i,f)|^5\right)^{\frac{1}{5}} \tag{5.34}$$

The $\delta s^t(f)$ values for each frame f are finally pooled using the power-3 Minkowski sum to yield the global speed based perceptual distance, $SPPD(\mathcal{M}_d, \tilde{\mathcal{M}}_d)$, as given by

$$SPPD(\mathcal{M}_d, \tilde{\mathcal{M}}_d) = \left(\frac{1}{N_f}\sum_{i=1}^{N_f} \left|\delta s^t(f)\right|^3\right)^{\frac{1}{3}} \tag{5.35}$$

(iii) **Angle-based perceptual distance (APD)**

The angle-based perceptual distance (APD) between $\mathcal{M}_d$ and $\tilde{\mathcal{M}}_d$ is based on the vertex-wise change in the motion directions across the frames of the mesh sequences. The forward and backward motion vectors are computed first for each vertex v_i^f in the frame f of the original mesh $\mathcal{M}_d$ and then the angle $\alpha_i^f \in [0,\pi)$ between the forward and the backward motion vectors is computed. Similarly, the angle $\tilde{\alpha}_i^f \in [0,\pi)$ between the forward and the backward motion vectors for each vertex $\tilde{v}_i^f$ in the frame f of the distorted mesh sequence is computed. The angle-based local perceptual distance $\delta a(i,f)$ between v_i^f and $\tilde{v}_i^f$ is

computed using the angle values α_i^f and $\tilde{\alpha}_i^f$ and given by

$$\delta a(i,f) = \frac{\left|\alpha_i^f - \tilde{\alpha}_i^f\right|}{\alpha_i^f + \tilde{\alpha}_i^f + C_a}, \quad i \in \{1,\ldots,N_v\}\,,\ f \in \{1,\ldots,N_f\}\,, \tag{5.36}$$

where $C_a = (\frac{1}{32})\pi$ is a small constant to avoid numerical instability. The derivation of $\delta a(i,f)$ also takes into account the temporal visual masking effect similar to $\delta s(i,f)$. The $\delta a(i,f)$ values for all the vertices in each frame are pooled spatially using the power-5 Minkowski sum to get the frame-wise local angle based perceptual distance $\delta a^t(f)$ as

$$\delta a^t(f) = \left(\frac{1}{N_v}\sum_{i=1}^{N_v}|\delta a(i,f)|^5\right)^{\frac{1}{5}} \tag{5.37}$$

The $\delta a^t(f)$ values for each frame f are finally pooled using the power-3 Minkowski sum to yield the global angle based perceptual distance, $APD(\mathcal{M}_d, \tilde{\mathcal{M}}_d)$, as

$$APD(\mathcal{M}_d, \tilde{\mathcal{M}}_d) = \left(\frac{1}{N_f}\sum_{i=1}^{N_f}\left|\delta a^t(f)\right|^3\right)^{\frac{1}{3}} \tag{5.38}$$

The $SWSPD(\mathcal{M}_d, \tilde{\mathcal{M}}_d)$, $SPPD(\mathcal{M}_d, \tilde{\mathcal{M}}_d)$ and $APD(\mathcal{M}_d, \tilde{\mathcal{M}}_d)$ distances are finally combined using the weighted mean square to compute the DMPD metric as given by Equation 5.24. The values of the weights are empirically set as $w_1 = 0.4$, $w_2 = 0.5$ and $w_3 = 0.1$ for the experiments done in [71] to get improved correlation between the DMPD metric and the subjective scores. The subjective evaluations in [71] were done on their self created database of dynamic 3D meshes from 10 reference animations impaired with different real-world and simulated distortions.

5.4 Proposed model-based PVD metric for 3D animations

Our work aims at investigating the performance of the existing PVD metrics, namely *STED* and *DMPD*, for the quality evaluation of 3D animations from the perceptual point of view and proposing a new PVD metric. An ideal model-based PVD metric for 3D animations must account for the visual error as interpreted by the HVS. We identify the following required properties that the PVD metric should satisfy in the context of its correlation with the HVS. Some of these properties are equally applicable to a PVD metric for the static 3D meshes. Our concern being the animation geometry compression, the proposed as well as compared quality metrics do not consider the effect of texture mapping on the perceptual quality of 3D objects.

Properties of a PVD metric

(i) **Invariance to affine transformations**

A 3D animation can be viewed from any viewpoint in the 3D coordinate system and the end-user's viewpoint information is not known a priori. The application of any kind of global affine transformations in terms of translation, scaling, rotation or their combinations on the set of vertices in the frames of a 3D animation alters its geometry attribute. But these affine transformation operations do not introduce any geometrical artifacts on the 3D mesh surface to be visually perceived by the end-users. Hence, a PVD metric for 3D animations should be invariant to any type of frame based affine transformations of the 3D objects about X, Y and Z axes. It means that if the frames of either of the original or the distorted 3D animations or both the animations are translated, scaled or rotated by some amount, the DM should not change its value from the human perceptual point of view. It is because the original and the distorted 3D animation are generally viewed independently by the end-users in separate 3D windows from any viewing angles. The viewing of an affine transformed or non-affine transformed 3D animation individually cannot perceptually detect the translation, the scaling or the rotation amount in the 3D mesh sequence, if any, introduced during the processing of the original mesh. The translation, scaling or rotation in the frames of either of the animations will be noticeable only if both the original and the transformed version of the decompressed 3D animations are viewed simultaneously within one viewing window from any angle. This is a very rare case of performance evaluation of the DMs. So, the final rule is that the PVD metric should detect and cancel the effect of any of the frame-based global translation, scaling or rotation in the distorted 3D animation with respect to the original animation to get the same value as with the non-transformation cases.

(ii) **Sensitivity to spatio-temporal artifacts**

The PVD metrics for 3D animations should detect any local artifacts that occur in the frames along the spatial and temporal directions. They should also take into account the visual masking effects sensitive to the HVS.

(iii) **Correlation with the opinion scores of human observers**

The PVD metrics should give output values having maximum correlation with the subjective opinion scores on the visual quality of the 3D animations by human observers.

Limitations of the STED and the DMPD metrics

The $STED$ and the $DMPD$ metrics satisfy the above properties to some extents. However, we have observed some issues with these metrics. They are discussed below.

a) The $STED$ metric value has been reported to show invariance to the global rigid body transformations in terms of translation and rotation operations applied on both the animations. However, it does not show invariance to the global scaling based affine transformation as indicated by our experimental results at the end of this chapter.

b) The $STED$ metric given in Equation (5.23) is a combination of spatial and temporal edge error terms with a weighting coefficient parameter c to be selected empirically. However, in the experimental result of the $STED$ metric, a value of $c = 9.144 \times 10^{-5}$ has been considered giving a negligible weight to the total temporal edge error term. This is to obtain maximum correlations with the subjective scores obtained for the distorted 3D animations. This raises the question about the need to compute the total temporal edge errors.

c) The $DMPD$ metric has been reported to be conceptually invariant to similarity transformation operations in terms of translation, rotation and scaling operations. However, our experimental results (shown in later section) suggest that the metric is not invariant to global rotation.

d) The $DMPD$ metric as described in Section 5.3.2 is a combination of three perceptual distance measures, the $SWSPD$, the $SPPD$ and the APD weighted by constants w_1, w_2 and w_3 respectively. It also depends on a number of constant terms whose values are selected empirically based on experimental results to get better overall correlation with the subjective scores. The computation time of the $DMPD$ metric is higher compared to the $STED$ metric.

Proposed $AISTED$ PVD metric

In the proposed PVD metric, we have tried to incorporate the required properties of a PVD metric as follows.

(i) The proposed PVD metric has been designed to show invariance to the effect of the frame-based affine transformation operations namely, translation, scaling and rotation, applied either on the original or the reconstructed animation. A method is devised to find the frame-wise affine transformation parameters for the rotation, scaling and translation operations between the set of vertices in the original and the reconstructed animations. An inverse homogeneous transformation matrix is formed for each frame using the combination of obtained parameters. The set of vertices in each frame of the reconstructed animation is multiplied by the frame-wise transformation matrix to get the affine invariant set of vertices.

(ii) The proposed PVD metric uses the relative changes in spatial as well as temporal edge lengths between the original and the reconstructed animations to measure the distortions in a perceptual manner. The spatial error part is computed using the average of normalized unique spatial edge length errors computed at each frame of the sequences. A unique spatial edge is defined as the edge connecting two vertices (irrespective of the order of the vertices) and shared by two triangular faces. The proposed unique edge based computation of spatial error is faster compared to the spatial error computation step followed in the STED metric. The temporal part is computed as average of normalized virtual temporal edge length errors computed at each frame.

(a) Determining affine invariance parameters

The parameters of the similarity transformation between the two sets of vertices in the 3D space can be found by applying a least squares method as proposed in [100] for the applications in the field of computer vision.

Let $\mathbf{G}^f = \left\{ \mathbf{p}_{v_i}^f : \ \mathbf{p}_{v_i}^f = \left(v_{i_x}^f, v_{i_y}^f, v_{i_z}^f \right)^T \in \mathbb{R}^3, \quad i = 1, \ldots, N_v, \quad f = 1, \ldots, N_f \right\}$ be the set of N_v vertices in the f-th frame of the original mesh sequence $\mathcal{M}_d$ and $\tilde{\mathbf{G}}^f = \left\{ \tilde{\mathbf{p}}_{v_i}^f : \ \tilde{\mathbf{p}}_{v_i}^f = \left(\tilde{v}_{i\_x}^f, \tilde{v}_{i\_y}^f, \tilde{v}_{i\_z}^f \right)^T \right\}$ be the corresponding set of vertices in the f-th frame of the reconstructed mesh sequence $\tilde{\mathcal{M}}_d$. Let us consider that the set of vertices in $\tilde{\mathbf{G}}^f$ are obtained after applying a combination of similarity transformations of rotation, uniform scaling and translation on the set of vertices in $\mathbf{G}^f$.

If s is the uniform scaling parameter for the x, y and z co-ordinates, $\mathbf{R}$ is the 3D transformation matrix for the rotation operations about X, Y and Z axes and $\mathbf{t} = [t_x \ t_y \ t_z]^T$ is the translation vector

containing the translation parameters t_x, t_y and t_z with respect to x, y and z coordinates respectively, then a vertex $\tilde{\mathbf{p}}_{v_i}^f$ in $\tilde{\mathbf{G}}^f$ can be expressed using the similarity transformation parameters $\mathbf{R}$, $\mathbf{t}$ and s as

$$\begin{bmatrix} \tilde{v}_{x_i}^f \\ \tilde{v}_{y_i}^f \\ \tilde{v}_{z_i}^f \end{bmatrix} = s\mathbf{R} \begin{bmatrix} v_{x_i}^f \\ v_{y_i}^f \\ v_{z_i}^f \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{5.39}$$

Therefore,

$$\tilde{\mathbf{p}}_{v_i}^f = s\mathbf{R}\mathbf{p}_{v_i}^f + \mathbf{t} \tag{5.40}$$

In the homogeneous coordinate system, the similarity transformation operations for a combination of rotation, scaling and translation operations on a point in the 3D space can be represented by a 4×4 transformation matrix $\mathbf{T}$ as

$$\mathbf{T} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & t_x \\ m_{21} & m_{22} & m_{23} & t_y \\ m_{31} & m_{32} & m_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{pmatrix} [s\mathbf{R}]_{3\times3} & [\mathbf{t}]_{3\times1} \\ 0 \quad 0 \quad 0 & 1 \end{pmatrix}_{4\times4} \tag{5.41}$$

To apply the 3D similarity transformations in homogeneous coordinates, a vertex $\mathbf{p}_{v_i}^f = \left(v_{i_x}^f, v_{i_y}^f, v_{i_z}^f\right)^T$ in 3D space is represented by a 4D vector as $\mathbf{p}_{v_i(h)}^f = \left[v_{i_x}^f \; v_{i_y}^f \; v_{i_z}^f \; 1\right]^T$. The mean square error $\xi_f^2(\mathbf{R}, \mathbf{t}, s)$ between the two sets of vertices in $\mathbf{G}^f$ and $\tilde{\mathbf{G}}^f$ in terms of the similarity transformation parameters $\mathbf{R}$, $\mathbf{t}$ and s is given by

$$\xi_f^2(\mathbf{R}, \mathbf{t}, s) = \frac{1}{N_v} \sum_{i=1}^{N_v} \left\| \tilde{\mathbf{p}}_{v_i}^f - (s\mathbf{R}\mathbf{p}_{v_i}^f + \mathbf{t}) \right\|^2 = \frac{1}{N_v} \sum_{i=1}^{N_v} \left\| \tilde{\mathbf{p}}_{v_{i(h)}}^f - \mathbf{T}\mathbf{p}_{v_{i(h)}}^f \right\|^2 \tag{5.42}$$

The transformation parameters $\mathbf{R}$, $\mathbf{t}$ and s can be obtained by solving the minimization problem

$$(\widehat{\mathbf{R}}, \hat{\mathbf{t}}, \hat{s}) = \underset{\mathbf{R},\mathbf{t},s}{\operatorname{argmin}} \left[\frac{1}{N_v} \sum_{i=1}^{N_v} \left\| \tilde{\mathbf{p}}_{v_i}^f - (s\mathbf{R}\mathbf{p}_{v_i}^f + \mathbf{t}) \right\|^2 \right] \tag{5.43}$$

The solution to the above problem is given by the singular value decomposition (SVD) of the covariance matrix $\mathbf{Cov}_f$ between the vertex coordinates vectors in the original and the reconstructed frames [100]. The mean and the variance of the vertices of the frame f from the original and reconstructed mesh

sequence are given by

$$\boldsymbol{\mu}^f = \frac{1}{N_v}\sum_{i=1}^{N_v}\mathbf{p}_{v_i}^f, \tag{5.44}$$

$$\tilde{\boldsymbol{\mu}}^f = \frac{1}{N_v}\sum_{i=1}^{N_v}\tilde{\mathbf{p}}_{v_i}^f, \tag{5.45}$$

$$\sigma_f^2 = \frac{1}{N_v}\sum_{i=1}^{N_v}(\mathbf{p}_{v_i}^f - \boldsymbol{\mu}^f)^2, \tag{5.46}$$

$$\tilde{\sigma}_f^2 = \frac{1}{N_v}\sum_{i=1}^{N_v}(\tilde{\mathbf{p}}_{v_i}^f - \tilde{\boldsymbol{\mu}}^f)^2 \tag{5.47}$$

The covariance matrix $\mathbf{Cov}_f$ between the vertex coordinate vectors of the frame f of $\mathcal{M}_d$ and $\tilde{\mathcal{M}}_d$ is given by

$$\mathbf{Cov}_f = \frac{1}{N_v}\sum_{i=1}^{N_v}(\tilde{\mathbf{p}}_{v_i}^f - \tilde{\boldsymbol{\mu}}^f)(\mathbf{p}_{v_i}^f - \boldsymbol{\mu}^f)^T \tag{5.48}$$

The SVD of $\mathbf{Cov}_f$ is given by

$$\mathbf{Cov}_f = \mathbf{U_f}\boldsymbol{\Sigma}_\mathbf{f}\mathbf{W_f}^T, \quad \mathbf{U_f}\mathbf{U_f}^T = \mathbf{W_f}\mathbf{W_f}^T = \mathbf{I}, \quad \boldsymbol{\Sigma}_\mathbf{f} = \text{diag}\,(\lambda_i), \quad \lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0 \tag{5.49}$$

where $\boldsymbol{\Sigma}_\mathbf{f}$ is the diagonal matrix of the eigen values λ_1, λ_2, λ_3 of the covariance matrix $\mathbf{Cov}^f$. Define

$$\mathbf{S} = \begin{cases} \mathbf{I} & \text{if} \quad \det(\mathbf{U_f})\det(\mathbf{W_f}) = 1 \\ \text{diag}(1,1,-1) & \text{if} \quad \det(\mathbf{U_f})\det(\mathbf{W_f}) = -1, \end{cases} \tag{5.50}$$

Then, the optimum similarity transformation parameters are given by

$$\hat{s} = \frac{1}{\sigma_f^2}tr(\boldsymbol{\Sigma}_\mathbf{f}\mathbf{S}) \tag{5.51}$$

$$\widehat{\mathbf{R}} = \mathbf{U_f}\mathbf{S}\mathbf{W_f}^\mathbf{T} \tag{5.52}$$

$$\hat{\mathbf{t}} = \tilde{\boldsymbol{\mu}}^f - s\mathbf{R}\mu^f \tag{5.53}$$

Next the inverse homogeneous transformation matrix $\mathbf{T_r} \in \mathbb{R}^{4\times4}$ is formed using the parameters s, $\mathbf{R}$ and $\mathbf{t}$. The set $\tilde{\mathbf{G}}^f$ of reconstructed vertices of frame f is multiplied by $\mathbf{T_r}$ to get the set $\tilde{\mathbf{G}}_s^f$ of affine transformation invariant vertices of the same frame.Thus

$$\tilde{\mathbf{G}}_{s(h)}^f = \mathbf{T_r}\tilde{\mathbf{G}}_{(h)}^f \tag{5.54}$$

where $\tilde{\mathbf{G}}^f_{(h)} \in \mathbb{R}^{4\times N_v}$ is a matrix storing the reconstructed vertices of the frame f in the homogeneous coordinates and $\tilde{\mathbf{G}}^f_{s(h)} \in \mathbb{R}^{4\times N_v}$ is a matrix storing the affine transformation invariant reconstructed vertices of frame f in the homogeneous coordinates. The PVD metric is now computed from these reconstructed frames.

(b) Computing spatial edge errors

Suppose, the affine transformation invariant vertices in frame f of $\tilde{\mathcal{M}}_d$ are represented by the set $\tilde{\mathbf{G}}^f_s = \left\{ \tilde{\mathbf{p}}^f_{v_i(s)} : \tilde{\mathbf{p}}^f_{v_i(s)} = \left(\tilde{v}^f_{i\_x(s)}, \tilde{v}^f_{i\_y(s)}, \tilde{v}^f_{i\_z(s)} \right)^T \right\}$. Let E_s represent the set of unique edges $e_s(k)$ containing the each pair of distinct vertices i, j derived from the connectivity data C_d and given by

$$E_s = \{e_s(k) : e_s(k) = \{l, m\}\}, \quad k = 1, \ldots, N_e, \quad l, m \in \{1, \ldots, N_v\} \tag{5.55}$$

where $N_e = |E_s|$ is the total number of unique edges $e_s(k)$ obtained using the connectivity information and l and m are the indices of the interconnected vertices to form the k-th edge $e_s(k)$. The set E_s representing the unique edges is constant for all the frames of an isomorphic 3D animation.

Next, the lengths of unique spatial edges in the original and the reconstructed frame f, $el(e_s(k), f)$ and $\tilde{el}(e_s(k), f)$ respectively, are computed as

$$\begin{aligned} el(e_s(k), f) &= \left\| \mathbf{p}^f_{v_l} - \mathbf{p}^f_{v_m} \right\| \\ \tilde{el}(e_s(k), f) &= \left\| \tilde{\mathbf{p}}^f_{v_{l(s)}} - \tilde{\mathbf{p}}^f_{v_{m(s)}} \right\| \end{aligned} \tag{5.56}$$

The normalized spatial edge length errors between the edge lengths in the original and reconstructed frame is now calculated as

$$se_{err}(f) = \frac{\left\| el(e_s(k), f) - \tilde{el}(e_s(k), f) \right\|_f}{\left\| el(e_s(k), f) \right\|_f}, \quad k = 1, \cdots, N_e \tag{5.57}$$

The mean of spatial edge errors, MEE_s across all the frames is computed as

$$MEE_s = \frac{1}{N_f} \sum_{f=1}^{N_f} se_{err}(f) \tag{5.58}$$

(c) Computing temporal edge errors

A temporal edge is defined as the virtual line segment formed by connecting the positions of a vertex in two consecutive frames. There will be total N_v temporal edges between two consecutive

frames formed by the N_v vertices per frame. The temporal edge lengths $el(e_t(i), f)$ and $\tilde{el}(e_t(i), f)$ for the vertex v_i^f in f-th frame of the original and the reconstructed animations are calculated as

$$\begin{aligned} el(e_t(i), f) &= \left\| \mathbf{p}_{v_i}^f - \mathbf{p}_{v_i}^{f-1} \right\| \\ \tilde{el}(e_t(i), f) &= \left\| \tilde{\mathbf{p}}_{v_{i(s)}}^f - \tilde{\mathbf{p}}_{v_{i(s)}}^{f-1} \right\| \quad i = 1, \cdots, N_v, \ f = 2, \cdots, N_f \end{aligned} \tag{5.59}$$

The normalized errors between the temporal edge lengths in the original and reconstructed frame is calculated as

$$te_{err}(f) = \frac{\left\| el(e_t(i), f) - \tilde{el}(e_t(i), f) \right\|_f}{\left\| el(e_t(i), f) + \tilde{el}(e_t(i), f) \right\|_f}, \quad i = 1, \cdots, N_v \tag{5.60}$$

The mean of temporal edge errors, MEE_t across all but the first frames is computed as

$$MEE_t = \frac{1}{N_f - 1} \sum_{f=2}^{N_f} te_{err}(f) \tag{5.61}$$

(d) Computing $AISTED$ metric

The proposed affine invariant spatio-temporal edge difference (AISTED) metric is finally computed as the combination of the MEE_s and MEE_t is given by

$$AISTED = \sqrt{w {MEE_s}^2 + (1 - w) {MEE_t}^2} \tag{5.62}$$

where w is the weighting parameter. Noting that the spatial error MEE_s depends on the number of edges N_e and the temporal error MEE_t depends on the number of vertices N_v, a reasonable choice of the weighting factor is

$$w = \frac{N_e}{N_e + N_v} \tag{5.63}$$

5.5 Experimental results

In this section, we present the details of the two set experiments that are performed to demonstrate the performance of the proposed $AISTED$ metric. In the first experiment, the performance of the $AISTED$ metric is measured to show its invariance to any combination of affine transformation operations in terms of translation, rotation and global scaling. In the second experiment, the correlation performances of the proposed metric is evaluated on two publicly available 3D animation databases. The correlation performance is compared to a few existing DMs used for quality assessment of 3D

animations. The details about the two 3D subjective databases and the results of the experiments are presented below.

5.5.1 3D animation databases

The two 3D animation databases used for finding the correlation performance with subjective assessment scores are: (i) the UWB 3D animation database [101] and (ii) the GIPSA-LAB 3D animation database [102]. The above two databases include different kinds of distorted 3D animations. The subjective assessment scores given by the human observers for each animation in the two datasets are also available. These subjective scores serve as the ground-truth for benchmarking and comparing the performance of any objective model-based PVD metric. The details about the two 3D animation databases are discussed below.

(i) **UWB database**

The UWB database contains 4 reference 3D animations namely, "Chicken" (3030 vertices, 400 frames), "Dance" (7061 vertices, 201 frames), "Falling cloth" (9987 vertices, 200 frames) and "Mocap Dance" (14409 vertices, 263 frames) and 36 distorted animations prepared from 9 distorted versions of each of the four reference animations. The 9 distortions are applied from a set of 13 distortions having 6 random noise distortions (listed at 1 to 6), 3 smooth spatial and temporal distortions (listed at 7 to 9) and 4 real-world distortions (listed at 10 to 13). They are listed below.

List of distortions used in UWB database

1. Adding different noise values from Gaussian distribution to each vertex coordinate in each frame.
2. Adding the same noise value from a Gaussian distribution for a given vertex coordinate over all the frames.
3. Adding the same noise value from a Gaussian distribution to all the vertices in a frame.
4. Adding different noise values from a Gaussian distribution to each vertex coordinate in each frame. The noise deviation selected for each vertex is according to the length of the edges incident with the given vertex.

5. Adding different noise values from a Gaussian distribution to each vertex coordinate in each frame. The noise deviation selected for each vertex is according to the speed of the movement of the given vertex.
6. Adding different noise values from the uniform distribution to each vertex coordinate in each frame.
7. Adding a value $A_d \sin(v_{i_x}^f \omega)$ which signifies a smooth distortion of amplitude A_d and frequency ω to each coordinate of each vertex.
8. Adding a value $A_d \sin(f\omega)$ which signifies a smooth temporal shifting of the whole mesh to each coordinate of each vertex.
9. Adding a value $A_d \sin(v_{i_x}^f \omega)$ to each coordinate of each vertex, i.e., a sinusoidal distortion of amplitude A_d and frequency ω. The amplitude A_d has been selected for each vertex according to the speed of the movement of the given vertex.
10. Subjecting each mesh to the *connectivity driven dynamic mesh compression* (CODDYAC) algorithm [34] with a coarse quantization scheme.
11. Subjecting each mesh to the Coddyac compression scheme using a small number of basis trajectories.
12. Subjecting each mesh to the Dynapack [35] compression scheme.
13. Subjecting each mesh to the compression scheme by Alexa and Müller [7].

Table 5.1 lists the details of distortion types applied to each of the reference 3D animations in the UWB database.

Table 5.1: List of distorted animations in UWB 3D animation database

Dataset Name	Name of 3D animations with the applied distortion number			
	Chicken	Dance	Falling cloth	Mocap Dance
A	11	2	2	10
B	1	3	10	12
C	11	1	3	13
D	2	10	9	6
E	11	2	1	13
F	5	11	10	12
G	1	11	5	13
H	8	4	9	6
I	7	4	8	11

The subjective assessment of the 36 distorted animations were carried out by human observers giving the visual quality scores with reference to the original mesh sequence. The scores were collected in the scale of 0 to 10 where 0 mark is for the best quality and 10 mark is for the worst quality with reference to the original animation.

(ii) **GIPSA-LAB database**

GIPSA-LAB animation database contains 10 reference and 276 distorted 3D animations. The distorted animations are prepared by applying 10 types of distortions on the set of 10 reference animations with the properties as listed in Table 5.2.

Table 5.2: Attributes of 3D animations used in GIPSA-LAB database

Animation Name	Vertices	Frames
armadillo	40002	75
balls	73960	41
chicken	3030	322
chinchilla	4307	84
clothBall	46598	69
dinosaur	20218	152
dress	41057	82
elephant	42321	49
horse	8431	48
human	18890	162

For each distortion type, 3 levels corresponding to low, medium and high distortion intensities were generated and applied to each of the reference 3D animation. Table 5.3 lists the details of distortion types applied to each of the reference 3D animations in the GIPSA-LAB database.

Table 5.3: List of distorted animations in GIPSA-LAB database

Dataset Name	Number of distortions types applied per Animation Sequence									
	Armadillo	Balls	Chicken	Chinchilla	Cloth-ball	Dinosaur	Dress	Elephant	Horse	Human
Coddyac	3	3	3	3	3	3	3	3	3	3
FAMC-DCT	3	3	3	3	3	3	3	3	3	3
FAMC-Lifting	3	3	3	3	3	3	3	3	3	3
Uniform	3	3	3	3	3	3	3	3	3	3
Gaussian	3	3	3	3	3	3	3	3	3	3
SM1	6	0	6	0	0	6	0	0	0	6
TM1	0	6	6	0	0	6	0	0	6	0
SM2	6	0	6	0	0	6	0	0	0	6
TM2	0	6	0	0	6	0	0	6	6	0
Network	3	3	3	3	3	3	3	3	3	3

The applied distortions are classified into two broad categories: a) simulated distortions and b) real-world distortions. The simulated distortions include global distortions and descriptor-weighted distortions. The global distortion signifies the addition of random noise values from either the uniform or the Gaussian distribution to the vertex coordinates in each frame. The descriptor-weighted distortion corresponds to the addition of noise to each vertex coordinate, which is proportional or inversely proportional to the local surface roughness or to the vertex speed. The descriptor-weighted distortions are used to simulate and study the behaviour of the visual spatial masking (SM) and the visual temporal masking (TM) effects. Two different kinds of noises were applied:

(a) Adding the same noise value from a uniform distribution to a given vertex coordinate over all the frames weighted by a coefficient which is proportional or inversely proportional to the local surface roughness or to the vertex speed (rows SM1 and TM1 in Table 5.3).

(b) Adding the same noise value from a uniform distribution to all the vertices in a frame weighted by a coefficient which is proportional or inversely proportional to the local surface roughness or to the vertex speed (rows SM2 and TM2 in Table 5.3).

The real-world distortions were used by applying the lossy compression algorithms for dynamic 3D meshes, namely the MPEG4-FAMC algorithm [2] and CODDYAC algorithm [34]. The subjective assessment of the 276 distorted animations was conducted, where the human observers gave their opinion scores about the perceived quality of the distorted and the reference 3D animations. Two categories of experiments are devised to collect the human opinion scores: i) *with-user interaction* (WI) and ii) *without-user interaction* (WOI). For the WI experiment, 25 human observers could freely zoom, rotate or translate the animations to give their opinion scores. For the WOI experiment, 16 human observers gave their opinion scores by simply observing the animations from a fixed point like video tracks. The subjective scores for both of the WI and WOI experiments were collected in the *single stimulus* (SS) way by displaying a single 3D mesh sequence to the human observer instead of the *double stimulus* (DS) way where both reference and distorted 3D animations are displayed together. Accordingly, subjective scores for the reference animations were also collected along with the distorted animations. The subjective

scores were collected in the scale of 0 to 10 where 0 mark is for the worst quality and 10 mark is for the best quality, reflected from the perceived quality of the displayed 3D animations. The differential scores for both the WI and WOI experiments were computed by subtracting the raw score of each distorted animation from the raw score of the corresponding reference animation in the same session. Finally, the DMOS value for each of the distorted animation is computed by taking the average of the differential scores from the human observers. The detailed explanation about the subjective experiment environmental setup and the analysis of all the distortions used in the GIPSA-LAB database can be found in [71].

5.5.2 Performance analysis using affine transformations

In this experiment, the performance of the proposed $AISTED$ metric is tested by applying affine transformations across the frames of a 3D animation. We have also tested the performances of existing three DMs namely, the KG_{err}, the $STED$ and the $DMPD$. All the four distortion metrics use a reference 3D animation for their calculations. The affine transformations in terms of translation, rotation and global scaling are applied to all the set of vertices per frame. The transformations can be applied individually as well as in combinations on an animation. The distortions between the original and the affine transformed animations are evaluated by the above four DMs. The results obtained by applying each affine transformation operation individually are presented below.

(i) **Applying global translation**

In this case, the affine transformation in terms of global translation factors of t_x, t_y and t_z in the x, y and z co-ordinates respectively is applied to each vertex across all the frames of an animation sequence. Table 5.4 shows the values of DMs obtained for different sets of translation parameters for the "Chicken" and "Cow" sequence.

Table 5.4: Performance of DMs for translation operation

Sequence	Transformation	Parameter	KG_{err}	$STED$	$DMPD$	$AISTED$
Cow	Translation	t_x=1, t_y=2, t_z=3	3.91E+03	5.42E-14	1.27E-11	2.09E-13
Cow	Translation	t_x=10, t_y=10, t_z=30	3.91E+04	4.52E-13	9.73E-11	2.20E-12
Chicken	Translation	t_x=10, t_y=20, t_z=30	644.5108	6.21E-14	8.47E-09	1.25E-11

It is observed that the $STED$, the $DMPD$ and the $AISTED$ metrics obtained give negligible errors for each translation operation for both the "Cow" and the "Chicken" animations. It

indicates that the $STED$, $DMPD$ and $AISTED$ metrics are invariant to global translation operation. The KG_{err} values of 3.91×10^3, 3.91×10^4 and 644.5108 obtained for the three translation operations signify that KG_{err} is not invariant to translation operation.

(ii) **Applying global rotation**

In this case, the affine transformation in terms of global rotation angles of r_x, r_y and r_z degrees about the x, y and z axes respectively is applied to the set of vertices across all the frames of an animation. It is to be noted that the global rotation operation only affects the viewing angle of the original animation. It does not cause any structural deformation and also does not introduce any visual perceptual distortion on the surface of the mesh. The viewing angle of a 3D animation in 3D space is not always fixed and an observer has the flexibility to view the animation from any angle as per requirement. So, a change of the viewing angle of a 3D animation by any rotation operation should not be accounted as a perceivable distortion by a PVD metric. Table 5.5 shows the values of DMs obtained for different sets of rotation parameter for the "Chicken" and "Cow" animations.

Table 5.5: Performance of DMs for rotation operation

Sequence	Transformation	Parameter	KG_{err}	$STED$	$DMPD$	$AISTED$
Chicken	Rotation	r_x=1°, r_y=1°, r_z=1°	12.3188	1.11E-05	0.0055	9.44E-11
Chicken	Rotation	r_x=10°, r_y=10°, r_z=10°	118.3982	1.14E-05	0.1045	3.38E-11
Chicken	Rotation	r_x=100°, r_y=100°, r_z=100°	578.4987	1.13E-05	0.2521	6.96E-11
Cow	Rotation	r_x=100°, r_y=100°, r_z=100°	233.681	1.48E-04	0.0983	1.24E-14

It is observed that, for both the animations, the $STED$ and $AISTED$ metrics show negligible errors for each set of rotational angle parameters. It indicates that both the $STED$ and $AISTED$ metrics are invariant to the global rotation operation. However, it is found that the KG_{err} and $DMPD$ metrics are affected by the rotation operation. For the "Chicken" animation, the values of KG_{err} and $DMPD$ metrics are found to be increased from 12.3188 to 578.4987 and from 0.0055 to 0.2521 respectively as the rotation angle parameters are varied from 1° to 100°. Similar results are obtained for the "Cow" animation. This indicates that the KG_{err} and $DMPD$ metrics are not invariant to the rotation operation.

(iii) **Applying global scaling**

In this case, we have applied the affine transformation in terms of global scaling operation to the x, y and z coordinates of each vertex across all the frames of the animation sequence. It has been observed that global scaling operation does not introduce any noticeable perceptual distortion on the surface of the mesh. It only affects the overall size of 3D objects in the animation. The scaling operation also does not affect user's viewing experience, since the original and the scaled version of the animations are always viewed independently. The change of size should not be accounted as a perceivable distortion by a PVD metric. Table 5.6 shows the values of the four DMs obtained for different values of scaling factor (s) for the "Chicken" and the "Cow" animations.

Table 5.6: Performance of DMs for scaling operation

Sequence	Transformation	Parameter	KG_{err}	$STED$	$DMPD$	$AISTED$
Cow	Scaling	$s = 10$	1.24E+03	8.00E-04	1.66E-12	1.19E-14
Cow	Scaling	$s = 200$	2.73E+04	0.0177	9.82E-13	1.20E-14
Cow	Scaling	$s = 1000$	1.37E+05	0.0887	8.84E-13	1.13E-14
Chicken	Scaling	$s = 1000$	6.06E+05	0.0854	8.38E-09	6.44E-11

It is observed that, for the "Cow" animation, the values of the KG_{err} and the $STED$ metrics are found to be increased from 1.24×10^3 to 1.37×10^5 and from 8.00×10^{-4} to 0.0887 respectively as the scaling parameter is varied from $s = 10$ to $s = 1000$. Similar results are obtained for the "Chicken" animation with KG_{err} value of 6.06×10^5 and $STED$ value of 0.0854 for $s = 1000$. It indicates that both the KG_{err} and the $STED$ metrics do not show invariance to the scaling operation. However, the $DMPD$ and $AISTED$ metrics give negligible error values for both the animations for all the scaling parameters. It indicates that the $DMPD$ and $AISTED$ metrics are invariant to the global scaling operation.

5.5.3 Correlation performance of the proposed metric

In this experiment, the correlation performance of the proposed $AISTED$ metric is evaluated on the UWB and the GIPSA-LAB 3D animation databases. The correlation performances of the $AISTED$ metric are compared with those of the KG_{err}, $STED$ and the $DMPD$ metrics. The performance results in terms of SROCC(%) and PLCC(%) on the two the databases are presented below.

A) Correlation results on the UWB database

Table 5.7 lists the 36 distorted 3D animations from the UWB database along with the corresponding MOS values.

Table 5.7: The MOS and the DM values for each distorted animation in the UWB database

Name of distorted animation	MOS	Metric name			
		KG_{err}	$STED$	$DMPD$	$AISTED$
chicken_A	7.77	0.3504	0.3601	0.4171	0.1282
chicken_B	9.84	0.2987	0.4573	0.4817	0.1874
chicken_C	1.63	0.2972	0.0353	0.2406	0.0428
chicken_D	7.7	0.2965	0.4546	0.4134	0.1310
chicken_E	2.16	0.4718	0.0486	0.2678	0.0642
chicken_F	6.3	0.2874	0.2915	0.4318	0.0977
chicken_G	6.64	0.1493	0.2252	0.4133	0.1080
chicken_H	0.95	0.6325	0.0000	0.1051	0.0000
chicken_I	1.79	0.2938	0.0637	0.2700	0.0653
falling cloth_A	9.24	0.4394	0.0590	0.4786	0.0512
falling cloth_B	7.86	0.3338	0.0312	0.4417	0.0239
falling cloth_C	2.22	0.2782	0.0000	0.1474	0.0000
falling cloth_D	1.32	0.2932	0.0048	0.0982	0.0080
falling cloth_E	2.73	2.6157	0.0249	0.2264	0.0493
falling cloth_F	8.78	0.2701	0.0453	0.4823	0.0815
falling cloth_G	6	0.2773	0.0346	0.4522	0.0321
falling cloth_H	2.46	0.8446	0.0121	0.1881	0.0256
falling cloth_I	2.32	0.2859	0.0000	0.1326	0.0000
dance_A	8.93	0.5481	0.1704	0.4657	0.0911
dance_B	1.95	7.4316	0.0001	0.3195	0.0000
dance_C	3.58	0.2564	0.0805	0.4053	0.0427
dance_D	6.74	0.5492	0.1693	0.4728	0.0918
dance_E	9.56	0.6990	0.1786	0.4794	0.1008
dance_F	1.88	0.5636	0.0150	0.1718	0.0203
dance_G	2.02	3.1070	0.0471	0.2424	0.0710
dance_H	7.95	0.5203	0.1103	0.4354	0.0854
dance_I	6.35	0.3217	0.0684	0.3931	0.0534
MocapDance_A	9.64	1.7852	0.4139	0.4547	0.1004
MocapDance_B	5.36	0.8926	0.1781	0.4050	0.0495
MocapDance_C	3.42	7.6388	0.0303	0.2202	0.0436
MocapDance_D	8.6	1.7852	0.4139	0.4547	0.1004
MocapDance_E	2.32	3.8149	0.0203	0.1827	0.0269
MocapDance_F	8.82	1.7855	0.3495	0.4743	0.1028
MocapDance_G	1.74	1.8749	0.0120	0.1521	0.0157
MocapDance_H	5.62	0.8928	0.2007	0.3997	0.0491
MocapDance_I	1.1	2.0380	0.0246	0.2618	0.0243

The values of KG_{err}, $STED$, $DMPD$ and $AISTED$ metrics obtained for each distorted 3D animation are also shown. The animation sequence-wise SROCC(%) and the PLCC(%) values between the DMs and the corresponding MOS values are given in Table 5.8.

Table 5.8: Animation sequence wise SROCCs(%) and PLCCs(%) between the DMs and the subjective MOS values for the UWB database

Metric Name	"Chicken"		"Falling Cloth"		"Dance"		"MocupDance"		Overall Database	
	SROCC	PLCC	SROCC	PLCC	SROCC	PLCC	SROCC	PLCC	SROCC	PLCC
KG_{err}	-18.33	-53.05	0.00	-27.03	-28.33	-53.94	-55.23	-32.83	-13.10	-27.19
$STED$	95.00	**97.01**	91.67	91.79	**95.00**	**89.75**	92.05	**97.79**	74.06	71.24
$DMPD$	88.33	93.34	**95.00**	**96.56**	91.67	86.32	85.36	91.24	**88.61**	**90.32**
$AISTED$	**96.67**	94.96	81.67	70.14	88.33	82.08	**93.72**	97.30	73.32	74.84

We observe that $AISTED$ metric gives the highest SROCC(%) value for the "Chicken" and the "Mocap Dance" animation whereas the $STED$ and the $DMPD$ metrics give the highest SROCC(%) values for the "Falling cloth" and the "Dance" animation respectively. The $STED$ metric has shown the highest PLCC(%) values for the "Chicken", "Dance" and the "Mocap Dance" animations whereas the $DMPD$ metric has shown the highest PLCC(%) value for the "Falling cloth" animation. For the overall database, the $DMPD$ metric has shown the highest SROCC(%) and the PLCC(%) values. However, for the $AISTED$ metric, the overall PLCC(%) is better than the $STED$ metric and the overall SROCC(%) value is nearly equal to that of the $STED$ metric.

Figure 5.2 illustrates the scatter plots of the four DMs versus the subjective MOS values of the UWB database. The red line in each plot is the fitted curve between the objective metric and the MOS values using the logistic psychometric function (Equation (5.13)) which maps the objective DMs to the MOS values. The plots in Figure 5.2(a) and 5.2(b) show a weak coherence between the KG_{err} and the $STED$ metrics with the MOS values. However, the plots of the $AISTED$ and $DMPD$ metrics show better coherence compared to that of the $STED$ metric. The animation sequence-wise PLCCs(%) after logistic psychometric fitting of the objective DMs and the MOS values are listed in Table 5.9. The logistic psychometric fitting has improved the PLCCs(%) of the $AISTED$ metric for the "Dance" and "Mocap Dance" animations compared to the other DMs and has also improved the overall PLCC(%) of the $AISTED$ metric compared

to the $STED$ metric.

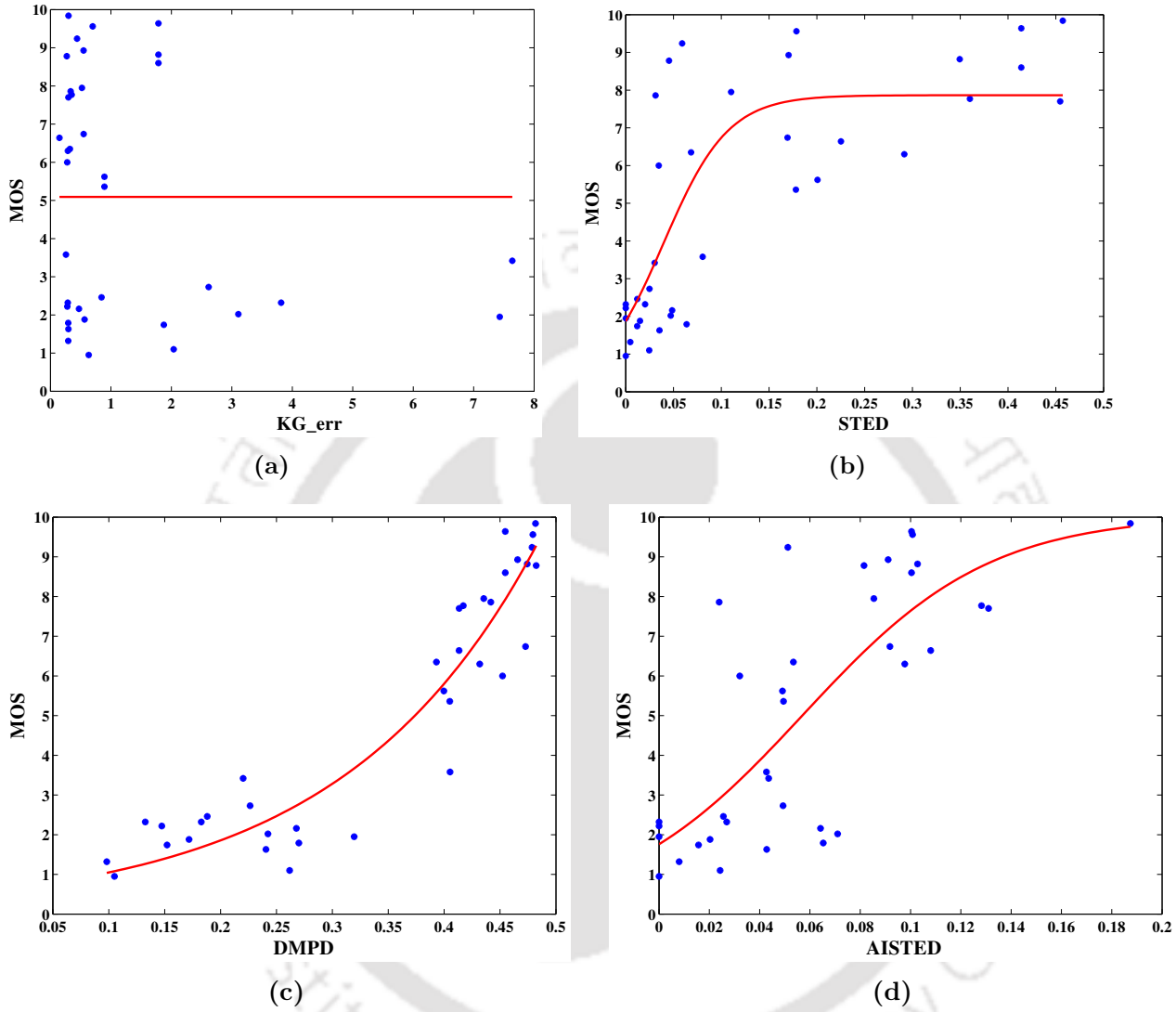


Figure 5.2: Scatter plots of the objective metrics versus the subjective MOS scores from UWB Dataset: (a) KG_{err}, (b) $STED$, (c) $DMPD$ and (d) $AISTED$ metrics. The red line in each plot is the fitted curve using the logistic psychometric function.

Table 5.9: Animation sequence-wise PLCCs(%) after the logistic psychometric fitting of the DMs for the UWB Database

Metric Name	"Chicken"	"Cloth"	"Dance"	"MocupDance"	Overall Database
KG_{err}	52.69	0.00	0.00	0.00	9.23
$STED$	88.14	**99.92**	95.56	88.80	84.45
$DMPD$	96.35	98.81	97.52	98.37	97.49
$AISTED$	**96.90**	99.68	**99.89**	**99.99**	**98.23**

B) **Correlation results on the GIPSA-LAB database**

The GIPSA-LAB animation database contains subjective scores in terms of the DMOS values for WI and WOI sessions as discussed earlier. The DMOS values for the WI session is suitable for finding the correlation between the objective DMs and subjective scores, since the WI session helps a human observer to judge a 3D animation in a better way using the interaction facilities like zoom, rotate or translate. However, in some of the subjective assessment tests like the UWB subjective database, 3D animations are displayed from a fixed viewpoint with the best frontal view of the 3D objects for the judgement by the human observers without any interaction facility. So, we have considered the DMOS values for both the WI and the WOI sessions for computing the SROCC(%) and PLCC(%) values of the proposed $AISTED$ metric using the GIPSA-LAB Database. In our experiment, we have considered the KG_{err}, $STED$, $DMPD$ and the $AISTED$ metrics to compute the distortion-wise SROCCs(%) and PLCCs(%) between the objective DMs and the DMOS values for the WI and the WOI sessions. The distortion-wise SROCCs(%) of the objective DMs computed with the DMOS values in the WOI and the WI sessions are listed in Table 5.10(a) and 5.10(b) respectively.

Table 5.10: Distortion-wise SROCCs(%) between the DMs and the subjective DMOS values for GIPSA-LAB database

(a) For the without-interaction(WOI) DMOS values

Metric	Category of Distorted Sequence										
	Coddyac	FAMC_Lifting	FAMC_DCT	TM1	SM1	TM2	Uniform	Gaussian	Network	SM2	Overall
KG_{err}	57.08	56.50	63.91	18.87	54.28	-35.15	41.30	39.94	52.03	86.16	23.00
$STED$	50.21	71.90	61.04	53.57	51.85	-23.02	64.44	50.07	54.93	74.85	56.65
$DMPD$	83.57	67.25	76.52	63.30	**85.30**	**59.00**	**89.30**	81.80	**80.73**	**90.69**	77.92
$AISTED$	**84.03**	**89.15**	**80.82**	**83.65**	**85.25**	53.82	83.98	**82.13**	75.25	81.64	**80.02**

(b) For the with-interaction(WI) DMOS values

Metric	Name of Distorted Sequences										
	Coddyac	FAMC_Lifting	FAMC_DCT	TM1	SM1	TM2	Uniform	Gaussian	Network	SM2	Overall
KG_{err}	47.35	45.87	50.98	34.37	42.14	-4.48	32.19	32.94	64.40	84.67	16.07
$STED$	45.34	67.43	58.54	68.57	67.10	12.53	45.90	47.61	62.22	85.98	57.64
$DMPD$	**86.59**	67.36	76.70	54.08	**91.50**	**90.04**	**81.41**	**79.75**	**88.54**	**95.72**	**78.80**
$AISTED$	78.85	**88.75**	**83.38**	**73.44**	87.85	42.54	70.69	58.18	79.76	82.89	75.10

From the results, it is observed that the proposed $AISTED$ metric gives the best SROCC(%) values of 84.03, 89.15, 80.82, 83.65 and 82.13 in the WOI session for the 3D animations having

distortions introduced from "Coddyac", "FAMC-Lifting", "FAMC-DCT", "TM1" and "Gaussian" operations respectively. The $DMPD$ metric gives the best SROCC(%) values of 85.30, 59.00, 89.30, 80.73 and 90.69 for the 3D animations with distortions introduced from the "SM1", "TM2", "Uniform", "Network and "SM2" operations respectively. The $AISTED$ metric gives the best overall SROCC(%) value of 80.02 for all the distortions in WOI session. For the DMOS values obtained in the WI session, it is observed that the proposed $AISTED$ metric gives the SROCC(%) values of 88.75, 83.38 and 73.44 for the 3D animations having distortions introduced from "FAMC-lifting","FAMC-DCT" and "TM1" operations respectively. The $DMPD$ metric gives the SROCC(%) values of 86.59, 91.50, 90.04, 81.14, 79.75, 88.54 and 95.72 for the 3D animations with distortions introduced from the "Coddyac", "SM1", "TM2", "Uniform", "Gaussion", "Network and "SM2" operations respectively. The $DMPD$ metric gives the overall SROCC(%) of 78.80 for all the distortions in WI session. However, the overall SROCC(%) of 75.10 obtained for the $AISTED$ metric is better than the overall SROCC(%) values of 57.64 and 16.07 for the $STED$ and the KG_{err} metrics respectively.

Similarly, the distortion-wise PLCC(%) values are computed for the WOI and WI sessions and listed in Table 5.11(a) and 5.11(b) respectively.

Table 5.11: Distortion-wise PLCCs(%) between the DMs and the subjective DMOS values for GIPSA-LAB database

(a) For the without-interaction(WOI) DMOS values

Metric	Category of Distorted Sequence										
	Coddyac	FAMC_Lifting	FAMC_DCT	TM1	SM1	TM2	Uniform	Gaussian	Network	SM2	Overall
KG_{err}	36.94	51.34	55.55	13.63	52.61	-25.88	35.80	29.03	32.62	43.49	2.69
$STED$	47.32	69.16	59.05	49.28	54.25	-25.51	65.46	53.07	54.75	51.43	55.30
$DMPD$	77.63	65.24	74.03	68.74	88.31	**60.14**	78.60	75.85	**78.65**	**87.62**	72.02
$AISTED$	**78.55**	**85.45**	**80.95**	**83.32**	**90.85**	43.54	**85.01**	**84.16**	72.18	81.66	**77.69**

(b) For the with-interaction(WI) DMOS values

Metric	Category of Distorted Sequences										
	Coddyac	FAMC_Lifting	FAMC_DCT	TM1	SM1	TM2	Uniform	Gaussian	Network	SM2	Overall
KG_{err}	31.26	43.60	46.78	36.86	40.15	8.67	22.92	24.76	52.22	39.49	4.74
$STED$	45.83	64.42	54.69	57.97	61.33	5.20	47.67	44.10	53.40	67.29	55.84
$DMPD$	**87.14**	66.65	75.13	53.74	**94.04**	**89.13**	66.70	**67.06**	**86.83**	**92.14**	**73.95**
$AISTED$	74.73	**85.31**	**82.65**	**77.56**	85.80	35.59	**71.88**	63.18	79.22	76.65	**73.11**

In this case, the $AISTED$ metric gives the best overall PLCC(%) value of 77.69 for all the distortion in the WOI session. For WI session, the overall PLCC(%) values obtained for $DMPD$ and $AISTED$ are ≈ 73.

Table 5.12(a) and 5.12(b) show the distortion-wise PLCCs(%) after logistic psychometric fitting of the DMs and the subjective DMOS values from the WOI and WI sessions respectively. The logistic fitting has improved the overall PLCC(%) of the $AISTED$ metric for both the WOI and WI sessions with values of 97.64 and 97.18 respectively.

Table 5.12: Distortion-wise PLCCs(%) after logistic psychometric fitting of the DMs with the subjective DMOS values of GIPSA-LAB database

(a) For the WOI DMOS values

Metric	Category of Distorted Sequence										
	Coddyac	FAMC_Lifting	FAMC_DCT	TM1	SM1	TM2	Uniform	Gaussian	Network	SM2	Overall
KG_{err}	36.54	52.88	46.73	38.97	88.00	0.00	38.80	47.54	62.24	-9.74	23.09
$STED$	**99.93**	**96.06**	98.04	75.50	93.85	-98.97	98.69	98.69	98.38	78.71	**97.55**
$DMPD$	89.49	95.42	91.73	**96.70**	**99.48**	**99.13**	92.94	95.12	**99.66**	**97.39**	96.79
$AISTED$	94.31	94.55	**99.37**	92.99	**99.50**	42.00	**98.97**	**98.97**	93.77	96.57	**97.64**

(b) For the WI DMOS values

Metric	Category of Distorted Sequence										
	Coddyac	FAMC_Lifting	FAMC_DCT	TM1	SM1	TM2	Uniform	Gaussian	Network	SM2	Overall
KG_{err}	36.62	43.19	40.10	47.22	89.63	0.00	37.84	43.62	61.47	38.43	21.36
$STED$	**99.72**	**96.63**	93.97	86.96	92.38	**99.98**	**99.02**	**92.66**	90.12	81.62	96.92
$DMPD$	96.14	94.47	92.79	**95.95**	**99.48**	98.86	89.42	87.97	**99.21**	96.25	96.26
$AISTED$	92.01	**96.24**	**99.01**	95.05	98.62	38.16	92.74	88.50	97.06	**98.23**	**97.18**

Figures 5.3 and 5.4 show the scatter plots of the four objective DMs versus the subjective DMOS values from the WOI and WI sessions respectively. The red line in each plot is the fitted curve between the DM and the DMOS values using the logistic psychometric function. The plots in Figures 5.3(a) and 5.4(a) show the weak coherence between the KG_{err} metric and the DMOS values for both the WOI and WI sessions. Similar performance is observed for the $STED$ metric. The logistic psychometric performance is found to be better for the $DMPP$ and AI_STED metrics for both the WOI and WI sessions. They are shown in Figures 5.3(c), 5.4(c) and 5.3(d), 5.3(d) respectively. The scatter plots show that the proposed $AISTED$ metric gives better correlation compared to the KG_{err} and $STED$ metrics. It gives similar performance as the $DMPD$ metric. However, the $AISTED$ metric involves

less computation as evident from the mathematical expressions. It may be good alternative as a distortion metric.

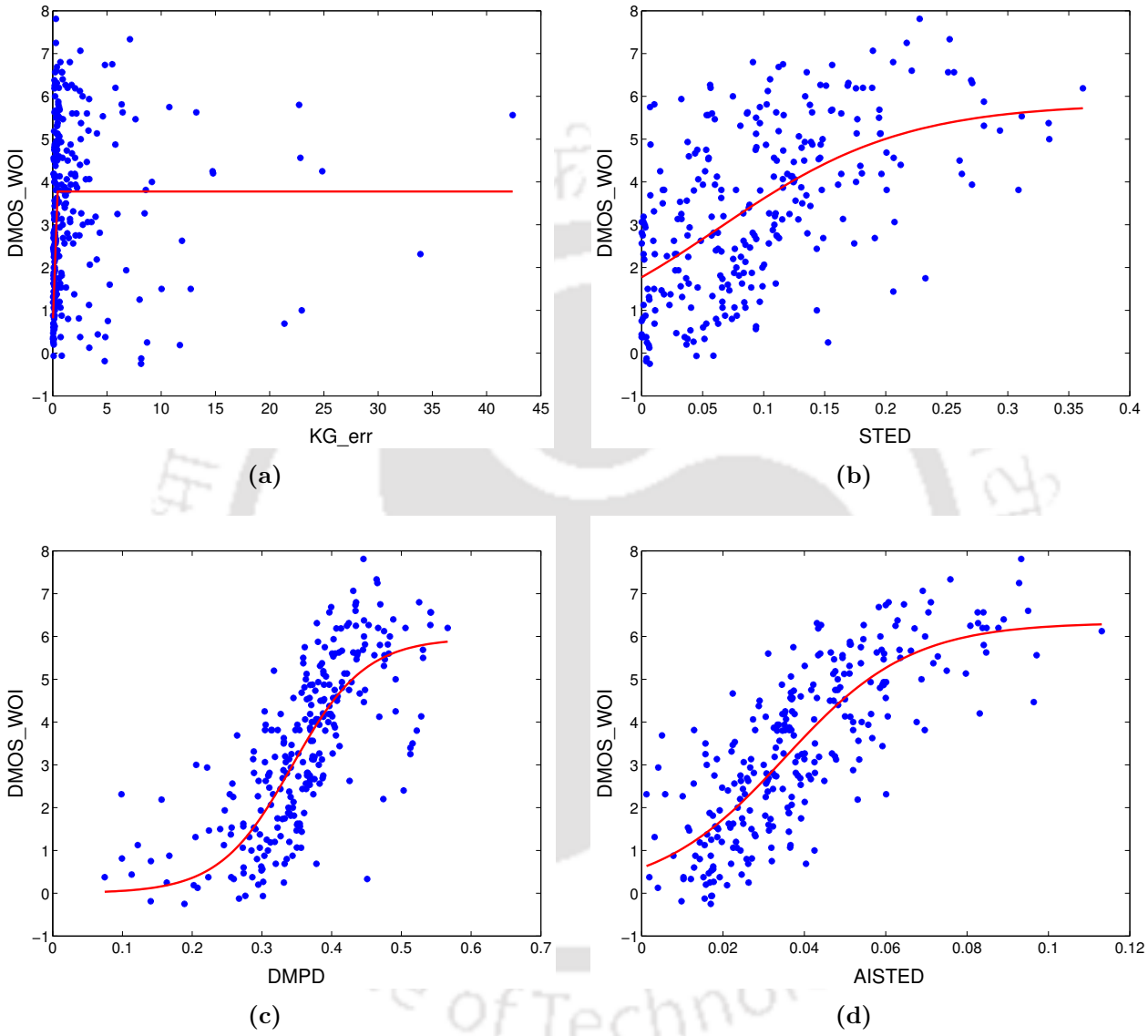


Figure 5.3: Scatter plots of the objective DMs (a) KG_{err}, (b) $STED$, (c) $DMPD$ and (d) $AISTED$ versus the subjective DMOS scores for the WOI session. The red line in each plot is the fitted curve using the logistic psychometric function.

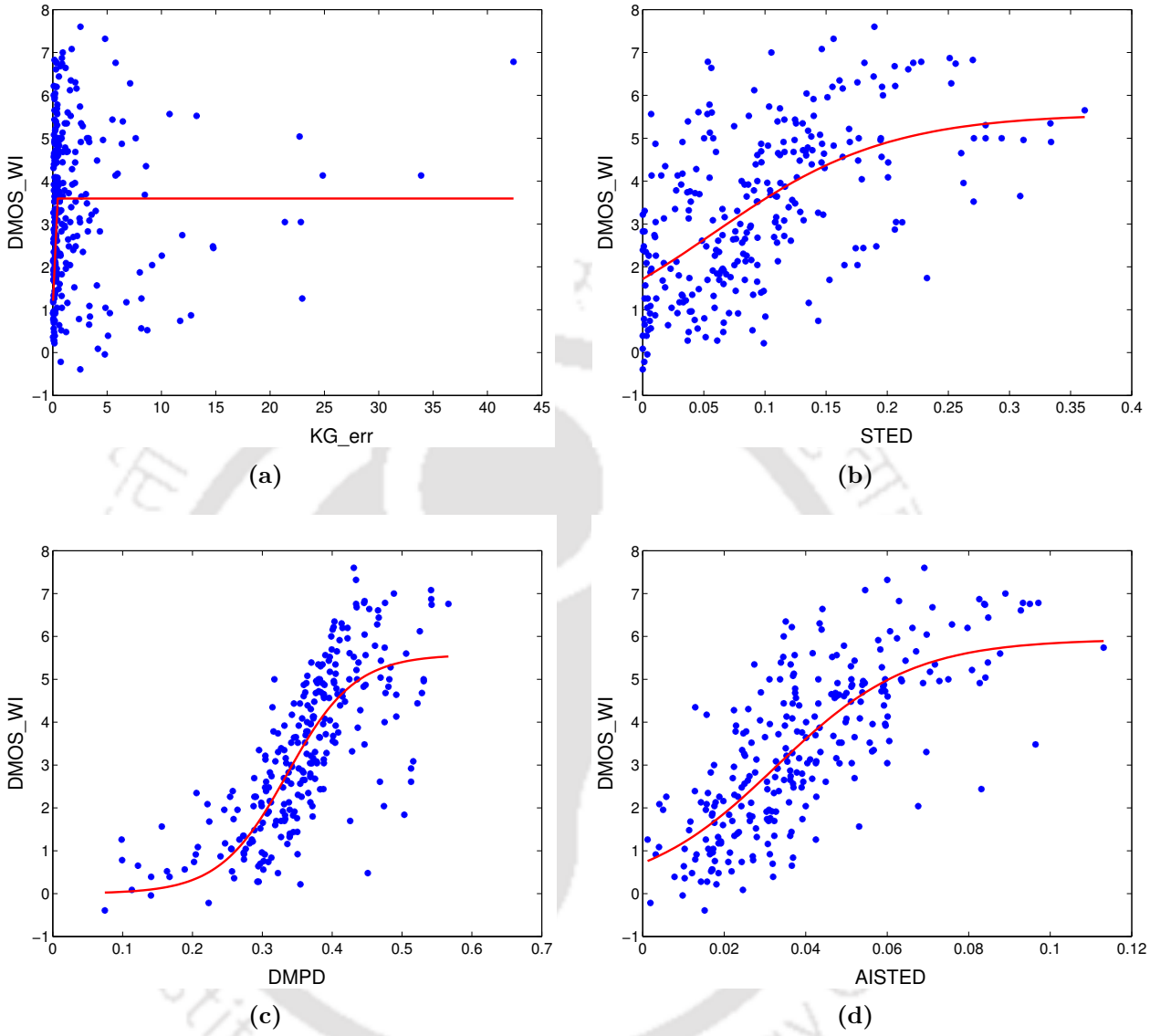


Figure 5.4: Scatter plots of the objective DMs (a) KG_{err}, (b) $STED$, (c) $DMPD$ and (d) $AISTED$ versus the subjective DMOS scores for the WI session. The red line in each plot is the fitted curve using the logistic psychometric function.

5.6 Conclusion

This Chapter presented a new perceptual visual distortion (PVD) metric, namely the $AISTED$ metric, for the quality assessment of 3D animations. The proposed $AISTED$ metric uses the relative changes in the spatial as well as the temporal edge lengths between the original and the reconstructed animations to measure the distortions in a perceptual manner. The $AISTED$ metric is designed to

show invariance to any combinations of frame-based affine transformation operations namely, translation, scaling and rotation, applied either on the original or the reconstructed animation. It uses a least-squares method in the 3D homogeneous coordinate system to find the frame-wise affine transformation parameters for the rotation, scaling and translation operations between the set of vertices in the original and the reconstructed animations. An inverse homogeneous transformation matrix, $\mathbf{T_r}$, is formed for each frame using the combination of obtained parameters. The set of vertices in each frame of the reconstructed animation is multiplied by the frame-wise transformation matrix to get the affine invariant set of vertices. After that, the normalized spatial and temporal edge length errors between the original and reconstructed frame are computed. The frame-wise spatial and temporal edge errors are averaged over all the frames to get the mean spatial edge error, MEE_s, and mean temporal edge error, MEE_t, respectively. The final $AISTED$ metric is computed as weighted combination of MEE_s and MEE_t. The correlation performances of the proposed $AISTED$ metric are evaluated with the subjective scores available on two public 3D animation databases: (i) UWB 3D animation database and (ii) GIPSA-LAB 3D animation database. The performances are compared with the existing three DMs namely, the KG_{err}, the $STED$ and the $DMPD$. For the UWB 3D animation database, the SROCC(%) of $AISTED$ metric for "Chicken" and "Mocup Dance" animations are found to be 96.67 and 93.72 respectively. This gives better correlations than all other metrics. The overall PLCC(%) of $AISTED$ on this database is found to be better than the $STED$ and KG_{err} metrics. For GIPSA-LAB 3D animation database, the proposed metric give best overall SROCC(%) and PLCC(%) values of 80.02 and 77.69 respectively compared to the other metrics when DMOS values are considered for the WOI case.

6

Conclusions and Future Work

Contents

The 3D animation geometry data correspond to the changing dynamics of the frame-wise vertex positions representing the 3D models across time. The research work in this thesis centred around the development of compression algorithms for these 3D animation geometry data. The main aim of our research was to compress the geometry data of isomorphic and block-isomorphic 3D animations for efficient storage and transmission over the heterogeneous network. This chapter summarizes the research contributions of thesis and points out some research directions.

6.1 Summary of contributions

The contributions from this thesis work can be divided into four parts. The first two parts deal with the non-scalable and scalable geometry compression of single-object isomorphic 3D animations. The third part considers the geometry compression of multi-object based block-isomorphic 3D animations. The last part proposes an improved distortion metric for the quality assessment of distorted 3D animations. These four contributions are outlined below.

(i) **Animation Geometry Compression using the Subtractive Clustering based CPCA**
A compression algorithm is proposed for improving the existing clustering based PCA (CPCA) method. The proposed method provides a stable initialization of cluster centres for the K-means clustering algorithm instead of a random initialization. The stable cluster centres are obtained by applying a density function based subtractive clustering algorithm on the vertex trajectories. The PCA is applied on each cluster to represent the clustered trajectories with a reduced number of PCs, the corresponding PC transformed coefficients and the mean trajectory vector. A non-uniform bit-allocation method is applied to quantize the elements of PCs and the PC coefficients per cluster. A block based arithmetic coding is proposed to encode the quantization levels of the PCs and PC coefficients quantized with same bits in each cluster. The compression performance of the proposed S-CPCA algorithm is evaluated in terms of the KG_{err} and the $STED$ distortion metrics for different target bitrates and compared with existing clustered based PCA methods. The proposed S-CPCA method performs better than other PCA based methods, namely the CPCA and the LPCA methods.

(ii) **Spatio-temporally Scalable Compression of Animation Geometry using SVD**

A novel scalable compression method with an encoder and a decoder structure has been proposed to obtain spatio-temporal scalability using the singular value decomposition (SVD) of the 3D animation geometry data. The method applies the SVD on the trajectory based animation geometry matrix **B** and selects a reduced number of spatial and temporal singular vectors based on the input threshold value of cumulative signal energy contained in the singular values. The set of elements in the spatial and temporal singular vectors are decomposed into different spatio-temporal layers to support scalability. To design the spatial layers, each frame is divided into sets of non-overlapping vertices using the patch based mesh simplification technique driven by the connectivity information. This spatial grouping information is used to arrange the elements of the spatial singular vectors into different spatial layers starting from a base layer to the full enhancement layer. The temporal layers are formed by arranging the elements of temporal singular vectors in a hierarchical manner from the base layer to the enhancement layers as followed in H.264 video coding. The prediction of elements in all the upper spatio-temporal layers is performed using the averaging of neighbourhood elements in the lower layers. The prediction errors of each spatio-temporal layer are quantized using a non-uniform bit allocation scheme and entropy coded using a block based arithmetic coding. The rate-distortion performance of the proposed method has been investigated on four standard 3D animation sequences in terms of the KG_{err} and the $STED$ metrics at different target bitrates. The performance has been also compared with three existing scalable compression methods. It is observed that the proposed compression algorithm outperforms these methods in terms of the scalable rates and distortions.

(iii) **Compression of Block-isomorphic Multi-object Animation Geometry**

This compression method exploits the spatio-temporal correlations of the individual 3D objects across the animation length. The method deals with the compression of the geometry data of block-isomorphic multi-object 3D (BIMO-3D) animations. The BIMO-3D animations contain multiple 3D objects per frame and satisfy the graph isomorphism property for a set of successive frames. The proposed method applies a temporal segmentation algorithm to divide the animation data of the consecutive frames into different isomorphic multi-object (IMO) blocks. Each IMO block satisfies the graph isomorphism property for the set of frames belonging to it. A proposed connectivity based spatial segmentation algorithm is applied on each IMO block to detect the

vertices belonging to the individual 3D objects. The geometry data of the individual 3D object are then separated out for each IMO block. The 3D objects spanning across multiple IMO blocks are detected based on their mesh attributes and the geometry data of those unique 3D objects are temporally merged across the frames in IMO blocks. The object based PCA (OPCA) algorithm is finally applied on the vertex trajectories of the merged unique 3D objects to get the compression. The compression performance of the proposed OPCA method is evaluated on a few BIMO-3D animations. The performance has been compared with the segment adaptive PCA (SA-PCA) and segment adaptive OPCA (SA-OPCA) methods which apply the PCA and OPCA algorithm respectively on geometry data of each IMO block without any merging operation. The compression performance of the merged OPCA (M-OPCA) is found to be better than that of the SA-PCA and the SA-OPCA methods.

(iv) **An Improved Distortion Metric for 3D Animation Geometry**

An improved perceptual visual distortion (PVD) metric, namely the $AISTED$ metric, is proposed for the quality assessment of 3D animations. The proposed $AISTED$ metric uses the relative changes in the spatial as well as the temporal edge lengths between the original and the reconstructed animations to measure the distortions in a perceptual manner. The $AISTED$ metric is designed to show invariance to any combinations of frame-based affine transformation operations namely, translation, scaling and rotation, applied either on the original or on the reconstructed animations. It uses a least-squares method in the 3D homogeneous coordinate system to find the frame-wise affine transformation parameters for rotation, scaling and translation operations between the set of vertices in the original and the reconstructed animations. An inverse homogeneous transformation matrix is formed for each frame using the combination of obtained parameters. The set of vertices in each frame of the reconstructed animation is multiplied by the frame-wise transformation matrix to get the affine invariant set of vertices. After that, the normalized spatial and temporal edge length errors between the original and reconstructed frame are computed. The frame-wise spatial and temporal edge errors are averaged over all the frames to get the mean spatial edge error, MEE_s and the mean temporal edge error, MEE_t, respectively. The final $AISTED$ metric is computed as the weighted combination of MEE_s and MEE_t. The correlation performances of the proposed $AISTED$ metric

are evaluated with the subjective scores available on two public 3D animation databases: (i) UWB 3D animation database and (ii) GIPSA-LAB 3D animation database. The performances are compared with the existing three DMs namely, the KG_{err}, the $STED$ and the $DMPD$. The correlation performances are measured in terms of the Pearson's linear correlation coefficient (PLCC) and the Spearman's rank order correlation coefficient (SROCC). For the UWB 3D animation database, the overall PLCC(%) of $AISTED$ metric is found to be better than the $STED$ and KG_{err} metrics for a few animations. For the GIPSA-LAB 3D animation database, the proposed metric give best overall SROCC(%) and PLCC(%) values compared to the other metrics when DMOS values are considered for the WOI case.

6.2 Future research directions

The contributions from this research work can be extended further to address some important research issues related to 3D animation geometry compression. A few of these future research directions are outlined below.

(i) Improving the compression performance of the S-CPCA method by finding the optimum number clusters for the given set of input parameters. The allocation of non-uniform quantization bits to each cluster can be further improved by exploring other statistical properties of the eigen values, eigen vectors and the PCA transformed coefficients.

(ii) The compression performance of the CPCA and LPCA algorithms may be further improved by applying a non-uniform quantizer and by varying the parameter c for each cluster. The implementation of the CPCA and LPCA algorithms with above modifications and comparing the performances with the proposed S-CPCA algorithm is a possible future research work.

(iii) In the SVD based scalable compression method, the design of spatial layers using the patch based mesh simplification technique does not result in smooth spatially scalable layers. The selection of centre vertex for patch decomposition should include some parameters related to the change in surface roughness. This may help in the enhancement of the visual perceptual quality of the lower spatial layers. The development of a new layer decomposition technique, therefore, is a possible research direction.

(iv) The elements in the higher spatio-temporal layers are predicted by averaging the neighbourhood elements present in the lower layer. The prediction method can be improved further by integrating some other linear or non-linear prediction methods to get better compression performance.

(v) The proposed SVD based scalable compression method supports only two types of scalability viz. spatial and temporal scalability. The integration of the quality scalability to the proposed scalable compression scheme may be carried out.

(vi) The proposed OPCA method deals with the non-scalable geometry compression of BIMO-3D animations. One of the ways to obtain scalable compression for BIMO-3D animation is by applying the SVD based scalable compression algorithm proposed in Chapter 3 on each isomorphic block. The scalable compression of BIMO-3D animation will be a challenging and an interesting extension of the current research work.

(vii) The compression performance of the OPCA method may be enhanced by applying clustering algorithms on the segmented large 3D objects.

(viii) The thesis addresses the geometry compression algorithms for isomorphic and block-isomorphic 3D animations. The development of geometry compression algorithms for non-isomorphic 3D animations having variable geometry and connectivity data in each frame may be a challenging future research direction.

(ix) Extending the AISTED metric to include the texture map, the frame rate and other psychovisual factors is a challenging research problem . The proposed $AISTED$ metric can be improved by incorporating other perceptual attributes like surface roughness and curvature information and local masking effects of the HVS.

(x) The correlation performance of the $AISTED$ metric with the subjective assessment scores may be carried out for the proposed geometry compression algorithms. For this, a database of 3D animations at various distortion levels has to generated by applying the proposed geometry compression algorithms on different 3D animations. Those distorted animations are to be evaluated subjectively by groups of human observer to record their perceptual quality in terms of MOS values and objectively in terms of the $AISTED$ metric.

Bibliography

[1] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, 2007.

[2] K. Mamou, N. Stefanoski, H. Kirchhoffer, K. Müller, T. Zaharia, F. Prêteux, D. Marpe, and J. Ostermann, "The new MPEG-4/FAMC standard for animated 3D mesh compression," in *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video, 2008*, May 2008, pp. 97 –100.

[3] Z. Karni and C. Gotsman, "Compression of soft-body animation sequences," *Computers and Graphics*, vol. 28, pp. 25–34, 2004.

[4] P. Rander, P. Narayanan, and T. Kanade, "Virtualized reality: constructing time-varying virtual worlds from real world events," in *Visualization '97., Proceedings*, Oct 1997, pp. 277–283.

[5] S. Vedula, P. Rander, H. Saito, and T. Kanade, "Modeling, combining, and rendering dynamic real-world events from image sequences," in *Proc. 4th Conference on Virtual Systems and Multimedia (VSMM98)*, November 1998, pp. 326 – 332.

[6] F. Yu, Z. Lu, H. Luo, and P. Wang, *Three-Dimensional Model Analysis and Processing.* Springer-Verlag Berlin Heidelberg, 2010.

[7] M. Alexa and W. Müller, "Representing animations by principal components," *Computer Graphics Forum*, vol. 19, no. 3, pp. 411–418, 2000.

[8] M. Sattler, R. Sarlette, and R. Klein, "Simple and efficient compression of animation sequences," in *ACM Symp. Computer Animation*, 2005, pp. 209–217.

[9] L. Váša and V. Skala, "COBRA: Compression of the basis for PCA represented animations," *Computer Graphics Forum*, vol. 28, no. 6, pp. 1529–1540, 2009.

[10] K. Müller, A. Smolic, M. Kautzner, and T. Wiegand, "Rate-distortion optimization in dynamic mesh compression," in *Image Processing, 2006 IEEE International Conference on*, oct. 2006, pp. 533 –536.

[11] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, Sept 1952.

[12] A. Moffat, R. M. Neal, and I. H. Witten, "Arithmetic coding revisited," *ACM Trans. Inf. Syst.*, vol. 16, no. 3, pp. 256–294, July 1998.

[13] H. Kirchhoffer, D. Marpe, K. Muller, and T. Wiegand, "Context-adaptive binary arithmetic coding for frame-based animated mesh compression," in *Multimedia and Expo, 2008 IEEE International Conference on*, June 2008, pp. 341–344.

[14] R. Amjoun and W. Strasser, "Predictive-DCT coding for 3D mesh sequences compression," *JVRB - Journal of Virtual Reality and Broadcasting*, vol. 5(2008), no. 6, 2008.

[15] L. Bo and Z. Hongbin, "Scalable mesh coding based on wavelet transform," in *Signal Processing, 2004. Proceedings. ICSP '04. 2004 7th International Conference on*, vol. 2, aug.-4 sept. 2004, pp. 1139 – 1142 vol.2.

[16] J.-H. Heu, C.-S. Kim, and S.-U. Lee, "SNR and temporal scalable coding of 3-D mesh sequences using singular value decomposition," *Journal of Visual Communication and Image Representation*, vol. 20, no. 7, pp. 439 – 449, 2009.

[17] Z. Karni and C. Gotsman, "Spectral compression of mesh geometry," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '00. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 2000, pp. 279–286.

[18] H. Hoppe, "Progressive meshes," in *ACM SIGGRAPH*, 1996, pp. 99–108.

[19] D. Cohen-Or, D. Levin, and O. Remez, "Progressive compression of arbitrary triangular meshes," in *Proceedings of the conference on Visualization*, 1999, pp. 67–72.

[20] A. Khodakovsky, P. Schröder, and W. Sweldens, "Progressive geometry compression," in *ACM SIGGRAPH*, 2000, pp. 271–278.

[21] R. Pajarola and J. Rossignac, "Compressed progressive meshes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 6, no. 1, pp. 79–93, jan 2000.

[22] G. Taubin and J. Rossignac, "Geometric compression through topological surgery," *ACM TRANSACTIONS ON GRAPHICS*, vol. 17, pp. 84–115, 1998.

[23] M. Deering, "Geometry compression," in *ACM SIGGRAPH*, 1995, pp. 13–20.

[24] M. Isenburg and P. Alliez, "Compressing polygon mesh geometry with parallelogram prediction," in *Proceedings of the conference on Visualization*, 2002, pp. 141–146.

[25] J. W. Cho, S. Valette, J. H. Park, H. Y. Jung, and R. Prost, "3-D mesh sequence compression using wavelet-based multi-resolution analysis," *Applied Mathematics and Computation*, vol. 216, no. 2, pp. 410 – 425, 2010.

[26] I. Guskov and A. Khodakovsky, "Wavelet compression of parametrically coherent mesh sequences," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2004, pp. 183–192.

[27] F. Payan and M. Antonini, "Temporal wavelet-based geometry coder for 3D animations," *Computer & Graphics*, vol. 31, pp. 77–88, 2007.

[28] S. Gumhold and W. Straber, "Real time compression of triangle mesh connectivity," in *ACM SIGGRAPH*, 1998, pp. 133–140.

[29] C. Touma and C. Gotsman, "Triangle mesh compression," in *Graphics Interface*, 1998, pp. 26–34.

[30] J. Rossignac, "Edgebreaker: Connectivity compression for triangle meshes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 1, pp. 47–61, Jan. 1999.

[31] J. E. Lengyel, "Compression of time-dependent geometry," in *Interactive 3D graphics*, 1999, pp. 89–95.

[32] S. Gupta, K. Sengupta, and A. A. Kassim, "Compression of dynamic 3D geometry data using iterative closest point algorithm," *Comput. Vis. Image Underst.*, vol. 87, no. 1-3, pp. 116–130, Jul. 2002.

[33] K. Mamou, T. Zaharia, and F. Prêteux, "A skinning approach for dynamic 3D mesh compression," *Comput. Animat. Virtual Worlds*, vol. 17, no. 3-4, pp. 337–346, July 2006.

[34] L. Váša and V. Skala, "CODDYAC: Connectivity driven dynamic mesh compression," in *3DTV Conference, 2007*, may 2007, pp. 1 –4.

[35] L. Ibarria and J. Rossignac, "Dynapack: space-time compression of the 3D animations of triangle meshes with fixed connectivity," in *ACM Symp. Computer Animation*, 2003, pp. 126–135.

[36] K. Müller, A. Smolic, M. Kautzner, P. Eisert, and T. Wiegand, "Predictive compression of dynamic 3D meshes," in *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, vol. 1, sept. 2005, pp. I – 621–4.

[37] N. Stefanoski and J. Ostermann, "Connectivity-guided predictive compression of dynamic 3D meshes," in *Image Processing, 2006 IEEE International Conference on*, oct. 2006, pp. 2973 –2976.

[38] N. Stefanoski, X. Liu, P. Klie, and J. Ostermann, "Scalable linear predictive coding of time-consistent 3D mesh sequences," in *3DTV Conference, 2007*, 2007, pp. 1–4.

[39] N. Stefanoski and J. Ostermann, "SPC: Fast and efficient scalable predictive coding of animated meshes," *Computer Graphics Forum*, vol. 29, no. 1, pp. 101–116, 2010, stefanoski, Nikolce Ostermann, Joern.

[40] J.-H. Yang, C.-S. Kim, and S.-U. Lee, "Semi-regular representation and progressive compression of 3-D dynamic mesh sequences," *Image Processing, IEEE Transactions on*, vol. 15, no. 9, pp. 2531 –2544, sept. 2006.

[41] F. Payan, A. Kamoun, and M. Antonini, "Remeshing and spatio-temporal wavelet filtering for 3D animations," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, 31 2008-april 4 2008, pp. 1081 –1084.

[42] N. Stefanoski and J. Ostermann, "Spatially and temporally scalable compression of animated 3D meshes with MPEG-4 / FAMC," in *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, oct. 2008, pp. 2696 –2699.

[43] R. Amjoun and W. Straßer, "Efficient compression of 3D dynamic mesh sequences." *Journal of WSCG*, vol. 15, no. 1-3, pp. 99–106, 2007.

[44] S.-R. Han, T. Yamasaki, and K. Aizawa, "Time-varying mesh compression using an extended block matching algorithm," *IEEE Trans. Cir. and Sys. for Video Technol.*, vol. 17, no. 11, pp. 1506–1518, Nov. 2007. [Online]. Available: http://dx.doi.org/10.1109/TCSVT.2007.903810

[45] T. Yamasaki and K. Aizawa, "Patch-based compression for time-varying meshes," in *Image Processing (ICIP), 2010 17th IEEE International Conference on*, sept. 2010, pp. 3433 –3436.

[46] M. O. Bici and G. B. Akar, "Improved prediction methods for scalable predictive animated mesh compression," *Journal of Visual Communication and Image Representation*, vol. 22, no. 7, pp. 577–589, 2011, bici, M. Oguz Akar, Gozde B.

[47] K. Mamou, T. Zaharia, and F. Prteux, "TFAN: A low complexity 3D mesh compression algorithm," *Computer Animation and Virtual Worlds*, vol. 20, no. 2-3, pp. 343–354, 2009.

[48] E.-Y. Chang, N. Hur, and E. S. Jang, "3d model compression in mpeg," in *2008 15th IEEE International Conference on Image Processing*, Oct 2008, pp. 2692–2695.

[49] L. Váša and V. Skala, "A perception correlated comparison method for dynamic meshes," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 17, no. 2, pp. 220–230, Feb 2011.

[50] J.-K. Ahn, Y. J. Koh, and C.-S. Kim, "Efficient fine-granular scalable coding of 3D mesh sequences," *Multimedia, IEEE Transactions on*, vol. 15, no. 3, pp. 485–497, 2013.

[51] L. Váša and V. Skala, "Geometry-driven local neighbourhood based predictors for dynamic mesh compression," *Computer Graphics Forum*, vol. 29, no. 6, pp. 1921–1933, 2010.

[52] L. Váša, "Optimised mesh traversal for dynamic mesh compression," *Graphical Models*, vol. 73, no. 5, pp. 218 – 230, 2011.

[53] S. P. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, pp. 129–137, 1982.

[54] G. Luo, F. Cordier, and H. Seo, "Compression of 3D mesh sequences by temporal segmentation," *Computer Animation and Virtual Worlds*, vol. 24, no. 3-4, pp. 365–375, 2013.

[55] S. Chiu, "Fuzzy model identification based on cluster estimation," *Journal of Intelligent Fuzzy Systems*, vol. 2, pp. 267–278, 1994.

[56] N. T. Ratrout, "Subtractive clustering-based k-means technique for determining optimum time-of-day breakpoints," *Journal of Computing in Civil Engineering*, vol. 25, no. 5, pp. 380–387, 2011.

[57] W.-Y. Liu, C.-J. Xiao, B.-W. Wang, Y. Shi, and S.-F. Fang, "Study on combining subtractive clustering with fuzzy c-means clustering," in *Machine Learning and Cybernetics, 2003 International Conference on*, vol. 5, Nov 2003, pp. 2659–2662 Vol.5.

[58] J. C. Bezdek, R. Ehrlich, and W. Full, "Fcm: The fuzzy c-means clustering algorithm," *Computers & Geosciences*, vol. 10, no. 2, pp. 191 – 203, 1984.

[59] M. El-Tarabily, R. Abdel-Kader, M. Marie, and G. Abdel-Azeem, "A pso-based subtractive data clustering algorithm," *International Journal of Research in Computer Science eISSN*, pp. 2249–8265, 2013.

[60] J. Heu, J.-H. Yang, C.-S. Kim, and S. Lee, "Effective quantisation scheme for principal components of 3-d mesh sequences," *Electronics Letters*, vol. 42, no. 14, pp. 799–800, 2006.

[61] F. Payan and M. Antonini, "An efficient bit allocation for compressing normal meshes with an error-driven quantization," *Computer Aided Geometric Design, Special Issue on Geometric Mesh Processing*, vol. 22, pp. 466–486, July 2005.

[62] Y. Boulfani-Cuisinaud, M. Antonini, and F. Payan, "Motion-based mesh clustering for MCDWT compression of 3D animated meshes," in *EUSIPCO'07*, Poznan, Poland, Sep. 2007, pp. 2105–2109.

[63] N. Stefanoski, P. Klie, X. Liu, and J. Ostermann, "Layered predictive coding of timeconsistent dynamic 3D meshes using a non-linear predictor," in *ICIP 2 Doi = 10.1109/3DTV.2007.4379461, 007. IEEE International Conference on*, vol. 5, 2007.

[64] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 7, pp. 620–636, July 2003.

[65] M. O. Bici and G. B. Akar, "Improved prediction for layered predictive animated mesh compression," in *Image Processing (ICIP), 2010 17th IEEE International Conference on*, sept. 2010, pp. 3413 –3416.

[66] H. Schwarz, D. Marpe, and T. Wiegand, "Analysis of hierarchical b pictures and MCTF," in *Multimedia and Expo, 2006 IEEE International Conference on*, July 2006, pp. 1929–1932.

[67] J.-K. Ahn, D.-Y. Lee, M. Ahn, J. Kim, J. Kim, and C.-S. Kim, "Progressive compression of 3D triangular meshes using topology-based karhunen-love transform," in *Image Processing (ICIP), 2010 17th IEEE International Conference on*, Sept 2010, pp. 3417–3420.

[68] C. Touma and C. Gotsman, "Triangle mesh compression," *PROC GRAPHICS INTERFACE. pp. 26-34. 1998*, 1998.

[69] K. Mamou, T. Zaharia, F. Prêteux, N. Stefanoski, and J. Ostermann, "Frame-based compression of animated meshes in MPEG-4," in *Multimedia and Expo, 2008 IEEE International Conference on*, 23 2008-april 26 2008, pp. 1121 –1124.

[70] K. Mamou, T. B. Zaharia, F. J. Prêteux, A. Kamoun, F. Payan, and M. Antonini, "Two optimizations of the MPEG-4 FAMC standard for enhanced compression of animated 3d meshes," in *ICIP*, 2008, pp. 1764–1767.

[71] F. Torkhani, K. Wang, and J. Chassery, "Perceptual quality assessment of 3D dynamic meshes: Subjective and objective studies," *Sig. Proc.: Image Comm.*, vol. 31, pp. 185–204, 2015.

[72] G. Lavoué and M. Corsini, "A comparison of perceptually-based metrics for objective evaluation of geometry processing," *IEEE Transactions on Multimedia*, vol. 12, no. 7, pp. 636–649, Nov 2010.

[73] M. Corsini, M.-C. Larabi, G. Lavoué, O. Petrík, L. Vása, and K. Wang, "Perceptual metrics for static and dynamic triangle meshes," *Computer Graphics Forum*, vol. 32, no. 1, pp. 101–125, Feb. 2013, improved version of Eurographics State-of-the-Art Report.

[74] A. Bulbul, T. Capin, G. Lavou, and M. Preda, "Assessing visual quality of 3-d polygonal models," *IEEE Signal Processing Magazine*, vol. 28, no. 6, pp. 80–90, Nov 2011.

[75] F. Torkhani, K. Wang, and A. Montanvert, "Towards perceptual quality evaluation of dynamic meshes," in *Proceedings of the ACM SIGGRAPH Symposium on Applied Perception in Graphics and Visualization*, ser. APGV '11. New York, NY, USA: ACM, 2011, pp. 116–116.

[76] Z. C. Yildiz and T. Capin, "A perceptual quality metric for dynamic triangle meshes," *EURASIP Journal on Image and Video Processing*, vol. 2017, no. 1, p. 12, 2017.

[77] S. Daly, "The visible differences predictor: An algorithm for the assessment of image fidelity," in *Digital Images and Human Vision*, A. B. Watson, Ed. Cambridge, MA, USA: MIT Press, 1993, pp. 179–206.

[78] J. Lubin, "A visual discrimination model for imaging system design and evaluation," *Vision models for target detection and recognition*, vol. 2, pp. 245–357, 1995.

[79] P. Lindstrom and G. Turk, "Image-driven simplification," *ACM Trans. Graph.*, vol. 19, no. 3, pp. 204–241, Jul. 2000.

[80] Z. Wang, H. R. Sheikh, and A. C. Bovik, "No-reference perceptual quality assessment of jpeg compressed images," in *Proceedings. International Conference on Image Processing*, vol. 1, 2002, pp. I–477–I–480 vol.1.

[81] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, April 2004.

[82] H. R. Sheikh, M. F. Sabir, and A. C. Bovik, "A statistical evaluation of recent full reference image quality assessment algorithms," *Trans. Img. Proc.*, vol. 15, no. 11, pp. 3440–3451, Nov. 2006.

[83] L. Zhang, L. Zhang, X. Mou, and D. Zhang, "Fsim: A feature similarity index for image quality assessment," *IEEE Transactions on Image Processing*, vol. 20, no. 8, pp. 2378–2386, Aug 2011.

[84] Z. Wang and Q. Li, "Information content weighting for perceptual image quality assessment," *IEEE Transactions on Image Processing*, vol. 20, no. 5, pp. 1185–1198, May 2011.

[85] G. Lavoué, M. C. Larabi, and L. Va, "On the efficiency of image metrics for evaluating the visual quality of 3d models," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 8, pp. 1987–1999, Aug 2016.

[86] B. E. Rogowitz and H. E. Rushmeier, "Are image quality metrics adequate to evaluate the quality of geometric objects," in *in Human Vision and Electronic Imaging*, 2001, pp. 340–348.

[87] P. Cignoni, C. Rocchini, and R. Scopigno, "Metro: Measuring error on simplified surfaces," *Computer Graphics Forum*, vol. 17, no. 2, pp. 167–174, June 1998.

[88] N. Aspert, D. Santa-Cruz, and T. Ebrahimi, "MESH: measuring errors between surfaces using the hausdorff distance," in *Multimedia and Expo, 2002. ICME '02. Proceedings. 2002 IEEE International Conference on*, vol. 1, 2002, pp. 705–708.

[89] F. Payan and M. Antonini, "Mean square error approximation for wavelet-based semiregular mesh compression," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 12, no. 4, pp. 649–657, July 2006.

[90] G. Lavoué, E. D. Gelasca, F. Dupont, A. Baskurt, and T. Ebrahimi, "Perceptually driven 3D distance metrics with application to watermarking," in *Proceedings of SPIE*, vol. 6312, 2006, p. 63120L.

[91] G. Lavoué, "A multiscale metric for 3d mesh visual quality assessment," *Computer Graphics Forum*, vol. 30, no. 5, pp. 1427–1437, 2011.

[92] L. Váša and J. Rus, "Dihedral angle mesh error: a fast perception correlated distortion measure for fixed connectivity triangle meshes," *Computer Graphics Forum*, vol. 31, no. 5, pp. 1715–1724, 2012.

[93] K. Wang, F. Torkhani, and A. Montanvert, "A fast roughness-based approach to the assessment of 3D mesh visual quality," *Computers & Graphics*, vol. 36, no. 7, pp. 808–818, 2012.

[94] F. Torkhani, K. Wang, and J.-M. Chassery, "A curvature tensor distance for mesh visual quality assessment," in *Proceedings of the International Conference on Computer Vision and Graphics - Volume 7594*, ser. ICCVG 2012. New York, NY, USA: Springer-Verlag New York, Inc., 2012, pp. 253–263.

[95] J.-W. Cho, M.-S. Kim, S. Valette, H.-Y. Jung, and R. Prost, "3-D dynamic mesh compression using wavelet-based multiresolution analysis," in *Image Processing, 2006 IEEE International Conference on*, oct. 2006, pp. 529 –532.

[96] J. Heu, J.-H. Yang, C.-S. Kim, and S.-U. Lee, "R-D optimized compression of 3-D mesh sequences based on principal component analysis," in *Proceedings of the 44th annual Southeast regional conference*, ser. ACM-SE 44. New York, USA: ACM, 2006, pp. 68–73.

[97] J.-H. Yang, C.-S. Kim, and S.-U. Lee, "Compression of 3-D triangle mesh sequences based on vertex-wise motion vector prediction," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 12, no. 12, pp. 1178–1184, Dec 2002.

[98] M. Corsini, E. D. Gelasca, T. Ebrahimi, and M. Barni, "Watermarked 3-d mesh quality assessment," *IEEE Transactions on Multimedia*, vol. 9, no. 2, pp. 247–256, Feb 2007.

[99] K. Seshadrinathan, R. Soundararajan, A. C. Bovik, and L. K. Cormack, "Study of subjective and objective quality assessment of video," *IEEE Transactions on Image Processing*, vol. 19, no. 6, pp. 1427–1441, June 2010.

[100] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 13, no. 4, pp. 376–380, Apr 1991.

[101] L. Vasa, *Data from STED Experiment*, 2011. [Online]. Available: http://meshcompression.org/data

[102] F. Torkhani and K. Wang, *GIPSA-lab 3D Mesh Animation Quality Database*, 10 2013. [Online]. Available: http://www.gipsa-lab.fr/~kai.wang/software/database/index.html

[103] A. Firouzmanesh, I. Cheng, and A. Basu, "Perceptually guided fast compression of 3-d motion capture data," *Multimedia, IEEE Transactions on*, vol. 13, pp. 829 – 834, 09 2011.

[104] O. Arikan, "Compression of motion capture databases," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 890–897, Jul. 2006. [Online]. Available: http://doi.acm.org/10.1145/1141911.1141971

[105] P. Beaudoin, P. Poulin, and M. van de Panne, "Adapting wavelet compression to human motion capture clips," in *Proceedings of Graphics Interface 2007*, ser. GI '07. New York, NY, USA: ACM, 2007, pp. 313–318. [Online]. Available: http://doi.acm.org/10.1145/1268517.1268568

[106] R. Mukundan, *Advanced Methods in Computer Graphics: with examples in OpenGL*, 1st ed., 1, Ed. Springer-Verlag London, 2012.

[107] Y. Pan, I. Cheng, and A. Basu, "Quality metric for approximating subjective evaluation of 3d objects," *Multimedia, IEEE Transactions on*, vol. 7, pp. 269 – 279, 05 2005.

[108] I. Cheng and P. Boulanger, "A 3d perceptual metric using just-noticeable-difference," pp. 97–100, 01 2005.

[109] I. Cheng and A. Basu, "Perceptually optimized 3-d transmission over wireless networks," *IEEE Transactions on Multimedia*, vol. 9, no. 2, pp. 386–396, Feb 2007.

[110] J. Guo, V. Vidal, I. Cheng, A. Basu, A. Baskurt, and G. Lavoue, "Subjective and objective visual quality assessment of textured 3d meshes," *ACM Trans. Appl. Percept.*, vol. 14, no. 2, pp. 11:1–11:20, Oct. 2016.

LIST OF PUBLICATIONS

1. Sanjib Das, P. K. Bora and A. K. Gogoi, "Subtractive Clustering of Vertices for CPCA based Animation Geometry Compression," *Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP2010)*, pp. 205-210, 12-15 Dec. 2010, Chennai, India.

2. Sanjib Das, J. Shah, P. K. Bora, "Temporally Scalable Compression of Animation Geometry," *Fourth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG 2013)*, 18-21 Dec. 2013, Jodhpur, India.

3. Sanjib Das, P. K. Bora, "Object-based Compression of 3D Animation Geometry," 12^{th} *International Conference on Signal Processing and Communications (SPCOM 2018)*, pp. 187-191, 16-19 July 2018, Bangalore, India.

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati